(d)

整理番号:98-05670

【書類名】　特許願
【整理番号】　9805670
【提出日】　平成11年 5月19日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 17/50
F16H 21/00
【発明の名称】　機構制御プログラム開発支援システム、機構制御プログラム開発支援装置、および機構制御プログラム開発支援プログラム記憶媒体
【請求項の数】　3
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　千田　陽介
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　佐藤　裕一
【発明者】
【住所又は居所】　神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
【氏名】　平田　光徳
【特許出願人】
【識別番号】　000005223
【氏名又は名称】　富士通株式会社
【代理人】
【識別番号】　100094330
【弁理士】
【氏名又は名称】　山田　正紀
【先の出願に基づく優先権主張】
【国際出願番号】　PCT/JP/98/05685
【出願日】　平成10年12月16日
【手数料の表示】
【予納台帳番号】　017961
【納付金額】　21,000円
【提出物件の目録】
【物件名】　明細書　1
【物件名】　図面　1
【物件名】　要約書　1
【包括委任状番号】　9704376
【プルーフの要否】　要

【書類名】　明細書
【発明の名称】　機構制御プログラム開発支援システム、機構制御プログラム開発支援装置、および機構制御プログラム開発支援プログラム記憶媒体
【特許請求の範囲】
【請求項１】　内部に構築された、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレータと、
前記三次元機構モデルシミュレータ内に構築された三次元機構モデルの動作を制御する制御プログラムを、前記三次元機構モデルシミュレータ内で動作する三次元機構モデルとの同期をとりながら実行する制御プログラムプロセッサとを備えたことを特徴とする機構制御プログラム開発支援システム。
【請求項２】　内部に構築された、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレータ内に構築された三次元機構モデルの動作を制御する制御プログラムを、前記三次元機構モデルシミュレータ内で動作する三次元機構モデルとの同期をとりながら実行することを特徴とする機構制御プログラム開発支援装置。
【請求項３】　アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレーションプログラムと、
前記三次元機構モデルの動作を制御する制御プログラムに付加されて、該制御プログラムを、三次元機構モデルシの動作との同期をとりながら実行させる同期化プログラムとを有する機構制御プログラム開発支援プログラムが記憶されてなることを特徴とする機構制御プログラム開発支援プログラム記憶媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、機構を制御する制御プログラムの開発を支援する機構制御プログラム開発支援システムおよび機構制御プログラム開発支援装置、および機構を制御する制御プログラムの開発を支援する機構制御プログラム開発支援プログラムを記憶してなる機構制御プログラム開発支援プログラム記憶媒体に関する。
【０００２】
【従来の技術】
近年、コンピュータグラフィックスとシミュレーション技術の発展で、コンピュータ内に三次元機構モデルを構築し、製品を試作することなくシミュレーションによってその三次元機構モデルの可動部を動かし、部品どうしが不用意に接触せず所期の動作を行なうことができるかどうか等、その機構の確認を行なうことができるようになってきており、その結果として、設計の不具合を早期に発見し、部品の配置変更や可動部を考慮した修正等を行なうことが可能になり、製品開発に要する時間の短縮化やコスト削減を図ることができるようになってきている。
【０００３】
ところで、そのような機構を動作させるには、そのような機構を制御する制御プログラムが開発され、その制御プログラムを実行することにより機構を動作させるのが一般的である。
【０００４】
ところが従来は、そのような制御プログラムを開発するには、例えば三次元機構モデルの動作シミュレーション等により動作上ほぼ完成した機構を物理的に試作し、その試作した機構を動作させながら制御プログラムの開発が行なわれている。このように、制御プログラムの開発、デバッグには試作品が必要であるため、試作品が完成するまでの間や試作品を修理、改造している間は制御プログラムの開発を行なうことができず、制御プログラムの開発に時間やコストがかかっていた。
【０００５】

【発明が解決しようとする課題】

本発明は、上記事情に鑑み、機構を制御する制御プログラムを容易に開発することができるように支援する機構制御プログラム開発支援システムおよび機構制御プログラム開発支援装置、および機構を制御する制御プログラムの開発を容易化する機構制御プログラム開発支援プログラムを記憶してなる機構制御プログラム開発支援プログラム記憶媒体を提供することを目的とする。

【０００６】

【課題を解決するための手段】

上記目的を達成する本発明の機構制御プログラム開発支援システムは、

内部に構築された、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレータと、

三次元機構モデルシミュレータ内に構築された三次元機構モデルの動作を制御する制御プログラムを、三次元機構モデルシミュレータ内で動作する三次元機構モデルとの同期をとりながら実行する制御プログラムプロセッサとを備えたことを特徴とする。

【０００７】

三次元機構モデルを構築しそれをシミュレーションで動作させる場合の動作速度は、その三次元機構モデルに相当する製品を試作し、その試作品を動作させる場合の動作速度とは一般には大きく異なっており、このままでは制御プログラムを実行させて三次元機構モデルを動作させても制御プログラムのデバックには不適当である。これに対し、本発明の機構制御プログラム開発支援システムを構成する制御プログラムプロセッサは、制御プログラムを、その制御プログラムにより制御される三次元機構モデルの動作との同期をとりながら実行するものであるため、三次元機構モデルを制御プログラムで制御しながら動作させてその動作を確認することによりその制御プログラムのデバッグを行なうことができる。

【０００８】

このように、本発明の機構制御プログラム開発支援システムによれば、試作品を製作することなく、三次元機構モデルをシミュレーションで動作させながら動作プログラムのデバッグを行なうことができるため、動作プログラムの開発に要する時間を短縮化し、開発コストを大きく低減することができる。

【０００９】

ここで、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルの動作と制御プログラムの実行との同期をとるにあたっては、

三次元機構モデルシミュレータが、制御プログラムプロセッサ内で動作する制御プログラムの実行時間を制御プログラムプロセッサに通知し、

制御プログラムプロセッサが、三次元機構モデルシミュレータからの制御プログラムの実行時間の通知を受けその実行時間だけ制御プログラムを実行して、その実行時間分の制御プログラムの実行の終了を三次元機構モデルシミュレータに通知し、

三次元機構モデルシミュレータが、制御プログラムプロセッサからの制御プログラムの実行時間分の実行の終了の通知を受け、三次元機構モデルを、制御プログラムのその実行時間分の制御に応じた動作量だけ動作させて、制御プログラムの次の実行時間を決定するものであってもよい。

【００１０】

例えばこのような手順を踏むことにより、三次元機構モデルの動作と制御プログラムの実行との同期をとることができる。

【００１１】

ここで、上記の手順を採用して同期をとるにあたっては、三次元機構モデルシミュレータが、制御プログラムプロセッサ内で動作する制御プログラムの実行時間を、三次元機構モデルを構成する部品どうしの間の距離に応じて決定するものであることが好ましい。

例えば部品どうしが大きく離れているときは、実行時間を大きくとって一回に

大きく動作させることにより所期の全体の動作を行なわせるまでの時間を短縮することができ、一方部品どうしが近づいたときは、実行時間を小さい値として少しずつ動かすことにより正確な動作確認を行なうことができる。

【００１２】

また、部品間の距離を求めるにあたっては、三次元機構モデルシミュレータが、動作プログラムの実行と連携した三次元機構モデルの動作に先立って、三次元機構モデルを構成する複数の部品を、三次元機構モデルを構成するアクチュエータごとに、そのアクチュエータの動作に応じて動作する部品の集合と、そのアクチュエータの動作によっては静止した状態にとどまる部品の集合とに分離し、動作プログラムの実行と連携した三次元機構モデルの動作にあたっては、これら分離した部品の集合どうしの間の、これら部品の集合どうしの間で常に干渉した状態にある部品間の距離を除く距離を求めるものであることが好ましい。

【００１３】

上記のように部品の集合に分け、それら部品の集合どうしの間の距離を求めることにより、距離を求めるための演算時間を大幅に短縮化することができる。

【００１４】

この場合に、さらに、三次元機構モデルシミュレータが、動作プログラムの実行と連携した三次元機構モデルの動作に先立って、その三次元機構モデルを構成するアクチュエータごとに、三次元機構モデルを構成する複数の部品のうちの、アクチュエータの動作に応じて動作する部品の集合を抽出し、そのアクチュエータを動作させることによりその集合を構成する部品を動作させてそれらその部品どうしの干渉の発生を検出し、動作プログラムの実行と連携した三次元機構モデルの動作にあたっては、常に干渉した状態にある部品を除く干渉が発生した部品どうしの間の距離を求めるものであることが好ましい。

【００１５】

上記のように部品の集合に分けたとき、アクチュエータの動作に応じて動作する部品の集合の中でも部品どうしが互いに干渉する可能性がある。このような場合に、その集合を構成する部品を動作させてそれらの部品どうしの干渉の発生を検出し、干渉の発生した部品どうしの間の距離を求めることにより、干渉の可能性のある部品どうしの間の距離を洩れなく求めることができる。

【００１６】

さらに、三次元機構モデルシミュレータが、三次元機構モデルを構成する部品間の距離を、その三次元機構モデルを構成するアクチュエータが動作している間のみ求めるものであることが好ましい。

【００１７】

距離を求める演算を必要な時のみに限ることにより、シミュレーションの高速化が図られる。

【００１８】

また、三次元機構モデルシミュレータ内で動作する三次元機構モデルと動作プログラムプロセッサ内で実行される動作プログラムとの同期をとるにあたっては、制御プログラムプロセッサ内で実行される制御プログラムが、少なくとも一部に、所定周期で繰り返し実行される部分を有し、その部分の実行回数に基づいて、その制御プログラムの実行時間分の実行の終了を認識するものであることが好ましい。

【００１９】

制御プログラムは、例えば一定時間ごとのタイマ割り込みで実行されるなど、所定周期で繰り返し実行される部分を有するのが一般的である。そこでこのような場合に、その部分が実行された回数を計数することによって制御プログラムの実行時間分の実行を認識することができる。

【００２０】

また、本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するアクチュエータを定

義するアクチュエータ定義手段を備え、該アクチュエータ定義手段が、該アクチュエータの定義にあたり、速度指令値と目標速度とを１：１で定義する有段変速と、速度指令値と目標速度との間の比例定数を定義する無段変速との双方の定義メニューを用意してなるものであることが好ましい。

【００２１】

このような２つの定義方法の双方を採用することにより、アクチュエータの定義の自由度が向上する。尚、ここでは‘有段変速’と‘無段変速’との２種類について述べたが、それらの‘有段変速’あるいは‘無段変速’の中でさらに複数種類の定義方法を採用してもよい。後述する実施形態では‘無段変速’の定義方法の中にさらに２種類の定義方法が採用されている。

【００２２】

また、本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータが、その三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するアクチュエータを一次遅れ系として動作させるものであることが好ましい。

【００２３】

アクチュエータを一次遅れ系として動作させることにより、そのアクチュエータの動作がほぼ一定の動作速度に達するまでの遅れ時間をあらわす整定時間を指定するだけでそのアクチュエータの立ち上がり時の振舞いを規定することができ、アクチュエータの動作の定義が容易となる。

【００２４】

さらに、本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するアクチュエータを定義するアクチュエータ定義手段を備え、そのアクチュエータ定義手段が、アクチュエータの動作により最初に動作する部品を、三次元機構モデルを三次元機構モデルシミュレータ内で動作させる際のアクチュエータとして定義するものであることが好ましい。

【００２５】

アクチュエータをこのように定義することで、三次元機構モデルの動作シミュレーションが容易となる。

【００２６】

さらに、本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するセンサを定義するセンサ定義手段を備え、そのセンサ定義手段が、センサにより検出される部品を、三次元機構モデルを該三次元機構モデルシミュレータ内で動作させる際のセンサとして定義するものであることが好ましい。

【００２７】

センサをこのように定義すると、上記のアクチュエータの定義と同様、シミュレーションが容易となる。

【００２８】

さらに、本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するセンサを定義するセンサ定義手段を備え、そのセンサ定義手段が、複数の部品間の干渉を検出することによりオン、オフを判定するセンサを定義するものであってもよい。

【００２９】

センサは、このような定義方法によってもシミュレーションを容易化することができる。

また、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するエンコーダを定義するエンコーダ定義手段を備え、三次元機構モデルシミュレータが、制御プログラムプロセッサからの制御プログラムの上記実行時間の実行の終了の通知を受け、三次元機構モデルを、制御プログラムのその実行時間分の制御に応じた動作

量だけ動作させた段階で、その実行時間分の制御に応じたエンコーダの動作量に応じたパルス数のパルスを出力するものであることが好ましい。

【００３０】

エンコーダを定義し、１回の実行時間分の制御量だけ動作させた段階でその実行時間分の制御に応じたエンコーダの動作量に応じたパルス数のパルスを出力することにより、エンコーダを正しくシミュレートすることができる。

【００３１】

ここで、上記エンコーダ定義手段が、エンコーダの出力パルスのパルス周波数の上限を設定する機能を有し、上記三次元機構モデルシミュレータが、エンコーダの動作量に応じたパルス数のパルスを出力するにあたり、設定されたパルス周波数の上限以下のパルス周波数のパルスを出力するものであることが好ましい。

【００３２】

現実のエンコーダの出力パルスはある一定のパルス周波数以下の周波数を持っており、パルス周波数の上限を設定しその上限以下のパルス周波数のパルスを出力することにより、より正確なシミュレートが可能となる。

【００３３】

また、本発明の機構制御プログラム開発支援システムにおいて、上記三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するポテンショメータを定義するポテンショメータ定義手段を備えたものであることが好ましい。

【００３４】

ポテンショメータも三次元機構モデルの１つの重要な構成要素となり得るからである。

【００３５】

また、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するセンサを定義するセンサ定義手段を備え、そのセンサ定義手段が、センサの状態を変化させる原因の発生からセンサの状態の実際の変化までの遅れを定義する手段を含むものであることが好ましい。

【００３６】

制御プログラムの開発にあたっては、アクチュエータやセンサが異常動作したときにも、暴走せずにあるコントロールされた状態を保つことが好ましく、そのような制御プログラムの開発にあたっては、アクチュエータやセンサの異常動作をシミュレートする必要がある。

本発明の機構制御プログラム開発支援システムがセンサ定義手段を備え、そのセンサ定義手段が、センサの状態をオンあるいはオフからオフあるいはオンに変化させる原因の発生からそのセンサの状態の実際の変化までの遅れを定義する手段を含むものであると、その遅れを定義し、センサの応答遅れという異常をシミュレートすることができる。

【００３７】

また、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するセンサを定義するセンサ定義手段を備え、そのセンサ定義手段が、センサの状態の変化に伴うチャッタリングを定義する手段を含むものであることが好ましい。

【００３８】

チャッタリングも、センサの、よく発生する異常動作の１つであり、このチャッタリングをシミュレートすることも重要である。

【００３９】

さらに、上記本発明の機構制御プログラム開発支援システムは、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成するセンサを定義するセンサ定義手段を備え、そのセンサ定義手段が、センサの故障を定義する手段を含むものであることが好ましく、また、上記本発明の機構制御プログラム開発支援システムは、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構

成するアクチュエータを定義するアクチュエータ定義手段を備え、そのアクチュエータ定義手段が、アクチュエータの故障を定義する手段を含むものであることが好ましい。

【００４０】

センサやアクチュエータの故障をシミュレートすることも重要である。

【００４１】

三次元機構モデルは数学的なモデルであるから、２つの部品どうしが重なっていても自由に動かすことができるが、実際の部品の場合は部品どうしが干渉するとそれ以上動作させることができない事態が生じる。したがって、上記本発明の機構制御プログラム開発支援システムにおいて、上記三次元機構モデルシミュレータは、三次元機構モデルを動作させている途中でその三次元機構モデルを構成する部品どうしの間に干渉が生じた場合に、干渉が生じた部品の動作を停止するものであることが好ましい。

【００４２】

また上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成する部品のうちのいずれかの部品を指定してその部品の動作を禁止する不良部品定義手段を備えることが好ましい。

【００４３】

センサやアクチュエータのみでなく、通常の従動部品も故障して動作不良となることもあり、それをシミュレートすることも重要である。

【００４４】

その場合に、上記三次元機構モデルシミュレータが、三次元機構モデルを動作させている途中における上記不良部品定義手段により指定された部品の指定を受けて、指定された部品の現在の動作方向への動作を禁止するものであることが好ましく、上記三次元機構モデルシミュレータが、上記不良部品定義手段による部品の、動作が禁止された方向とは逆の方向への動作に応じて、その部品の、動作が禁止された方向についての動作の禁止を解除するものであることが好ましい。

【００４５】

このようにして、何かに引っ掛かって動作不良となり、逆方向に動かすことにより正常に戻る状態をシミュレートすることができる。

【００４６】

また、上記本発明の機構制御プログラム開発支援システムにおいて、上記三次元機構モデルシミュレータが、三次元機構モデルを構成する部品の可動範囲を、その部品を動作させたときのその部品と他の部品との間の干渉を検出することにより自動設定する可動範囲設定手段を有するものであることが好ましい。

【００４７】

干渉検出の手法を用いて部品の可動範囲を自動設定することにより、その可動範囲をオペレータがいちいち入力する手間が省かれ、使い勝手の良い、操作性に優れたシステムが実現する。

【００４８】

さらに、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータが、三次元機構モデルを構成する２つの部品のカム関係を、これら２つの部品のうちの一方の部品を動作させたときのこれら２つの部品の間の干渉を検出することにより自動設定するカム関係設定手段を有するものであることが好ましい。

【００４９】

カム関係にある２つの部品の連携動作に関しても、干渉検出の手法により自動設定することにより、オペレータの手間が省かれ、使い勝手の良いシステムが実現する。

【００５０】

カム関係を自動設定するにあたっては、２つの部品のうちの一方がバネや重力

により一方向に付勢されたものである場合がある。その場合、カム関係設定手段が、上記２つの部品のうちの他方の部品が所定方向に付勢されたものである場合の、これら２つの部品の間の干渉を検出することにより、これら２つの部品のカム関係を自動設定するものであることが好ましい。

【００５１】

さらに、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータが、三次元機構モデルを構成する２つの部品のうちの一方の部品の溝に他方の部品のピンが挿入された溝関係を、そのピンに代えてそのピンよりも細径の仮想部品を作成しその仮想部品と溝壁との間の距離を検出することにより自動設定する溝関係設定手段を有するものであることが好ましい。

【００５２】

溝に挿入されたピンは、通常その溝壁に常に接触しており、このままでは非接触状態を作り出すことができない。そこで、上記のような細径の仮想部品を作成し、溝壁との間の短縮を検出することにより、溝関係を自動設定することができる。

【００５３】

さらに、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータが、三次元機構モデルを構成する部品を動作させるプログラムを再帰的に実行することにより三次元機構モデルを動作させるものであることが好ましい。

【００５４】

このような再帰的に実行されるプログラムを採用することにより、三次元機構モデルの動作シミュレーションの高速化が可能になる。

【００５５】

さらに、上記本発明の機構制御プログラム開発支援システムにおいて、三次元機構モデルシミュレータ内で動作する三次元機構モデルを構成する２つの部品の連携動作の有無を切り換えるクラッチを定義するクラッチ定義手段を備えたものであることが好ましい。

【００５６】

クラッチも、三次元機構モデルを構成する重要な部品の１つとなり得るため、これをシミュレートすることができるようにしておくことが好ましい。

【００５７】

また、上記目的を達成する本発明の機構制御プログラム開発支援装置は、内部に構築された、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレータ内に構築された三次元機構モデルの動作を制御する制御プログラムを、三次元機構モデルシミュレータ内で動作する三次元機構モデルとの同期をとりながら実行するものであることを特徴とする。

【００５８】

三次元機構モデルシミュレータが別に構成されているときは、機構制御プログラム開発支援装置を構成することにより、動作プログラムの開発支援を行なうことができる。

【００５９】

また、上記目的を達成する本発明の機構制御プログラム開発支援プログラム記憶媒体は、

アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレーションプログラムと、

三次元機構モデルの動作を制御する制御プログラムに付加されて、その制御プログラムを、三次元機構モデルの動作との同期をとりながら実行させる同期化プログラムとを有する機構制御プログラム開発支援プログラムが記憶されてなることを特徴とする。

【００６０】

本発明の機構制御プログラム開発支援プログラム記憶媒体に記憶された機構制御プログラム開発支援プログラムをコンピュータ内で実行することにより、シミュレーション上で動作プログラムのデバッグを行なうことができ、動作プログラムの開発が容易となる。

【００６１】

ここで、上記本発明の機構制御プログラム開発支援プログラム記憶媒体において、

三次元機構モデルシミュレートプログラムが、制御プログラムの実行時間を同期化プログラムに通知し、

同期化プログラムが、三次元機構モデルシミュレートプログラムからの、制御プログラムの実行時間の通知を受け、その実行時間分だけ制御プログラムを実行させて、その実行時間分の制御プログラムの実行の終了を三次元機構モデルシミュレートプログラムに通知し、

三次元機構モデルシミュレートプログラムが、同期化プログラムからの、制御プログラムの上記実行時間分の実行の終了の通知を受け、三次元機構モデルを、制御プログラムのその実行時間分の制御に応じた動作量だけ動作させて、制御プログラムの次の実行時間を決定するものであることが好ましい。

【００６２】

たとえばこのような通知のやりとりにより、三次元機構モデルの動作と制御プログラムの実行との間で同期をとることができる。

【００６３】

【発明の実施の形態】

以下、本発明の実施形態について説明する。

【００６４】

図１は、内部に機構制御プログラム開発支援システムが構築されたコンピュータシステムの外観図である。

【００６５】

このコンピュータシステム１００は、ＣＰＵ、ＲＡＭ、ハードディスク等を内蔵した本体部１０１、本体部１０１からの指示により表示画面１０２ａに画面表示を行うＣＲＴディスプレイ１０２、このコンピュータシステムにユーザの指示や文字情報を入力するためのキーボード１０３、表示画面１０２ａ上の任意の位置を指定することによりその位置に表示されていたアイコン等に応じた指示を入力するマウス１０４を備えている。

【００６６】

本体部１０１は、さらに、外観上、フロッピィディスク２１２（図１には図示せず；図２参照）やＣＤＲＯＭ２１０が取り出し自在に装填されるフロッピィディスク装填口１０１ａおよびＣＤＲＯＭ装填口１０１ｂを有しており、その内部には、装填されたフロッピィディスクやＣＤＲＯＭ２１０をドライブする、フロッピィディスクドライバ２２４、ＣＤＲＯＭドライバ２２５（図１０参照）も内蔵されている。

【００６７】

ここでは、ＣＤＲＯＭ２１０に、本発明にいう機構制御プログラム開発支援プログラムが記憶されている。このＣＤＲＯＭ２１０がＣＤＲＯＭ装填口１０１ｂから本体部１０１内に装填され、ＣＤＲＯＭドライバ２２５により、そのＣＤＲＯＭ２１０に記憶された機構制御プログラム開発支援プログラムがこのコンピュータシステム１００のハードディスク内にインストールされる。さらに、このコンピュータシステム１００のハードディスクには、ユーザによりキーボードやマウス等を用いて定義された三次元機構モデル、および、ユーザにより、やはりキーボードやマウス等を用いて入力された、その三次元機構モデルを実際に製造したときの製品を制御するための制御プログラムも格納されている。尚、三次元機構モデルや制御プログラムは、それらのうちのいずれか一方あるいは双方が、この図１に示すコンピュータ１００とは別のシステムで構築ないしプログラミング

されて、このコンピュータ１００に格納されたものであってもよい。

【００６８】

このコンピュータシステム１００内に構築された三次元機構モデルは、ＣＤＲＯＭ２１０からこのコンピュータシステム１００内にローディングされた機構制御プログラム開発支援プログラムを構成する三次元機構モデルシミュレーションプログラムによりそのコンピュータシステム１００内で動作し、一方、このコンピュータシステム１００内に組み込まれた動作プログラムは、ＣＤＲＯＭ２１０からこのコンピュータシステム１００内にローディングされた機構制御プログラム開発支援プログラムを構成する同期化プログラムと結合され、このコンピュータシステム１００内では、その動作プログラムが、三次元機構モデルの動作と同期して実行される。

【００６９】

ここでは、動作プログラムのデバッグを対象としており、したがってここでは、コンピュータシステム１００内に構築された三次元機構モデル自体は、動作プログラムとの連携動作の途中で不具合が発見されたときは修正されることはあるものの一応その動作が確認されたものであり、動作プログラムはデバッグ中のものであるとする。

【００７０】

ここでは、本発明との対応関係でいうと、このコンピュータシステム１００の、三次元機構モデルを動作させる機能が三次元機構モデルシミュレータに相当し、同期化プログラムと結合された動作プログラムを実行する機能が制御プログラムプロセッサおよび機構制御プログラム開発支援装置に相当する。

【００７１】

このように、本発明の機構制御プログラム開発支援システムには、１つの装置（ここではコンピュータシステム１００）内に三次元機構モデルシミュレータと動作プログラムプロセッサとの双方が組み込まれた形態も含まれる。

【００７２】

図２は、図１に外観を示すコンピュータシステムのハードウェア構成図である。

【００７３】

ここには、中央演算処理装置（ＣＰＵ）２２１、ＲＡＭ２２２、ハードディスクコントローラ２２３、フロッピィディスクドライバ２２４、ＣＤＲＯＭドライバ２２５、マウスコントローラ２２６、キーボードコントローラ２２７、およびディスプレイコントローラ２２８が示されており、それらはバス２２０で相互に接続されている。

【００７４】

フロッピィディスクドライバ２２４、ＣＤＲＯＭドライバ２２５は、図１を参照して説明したように、フロッピィディスク２１２、ＣＤＲＯＭ２１０が装填され、装填されたフロッピィディスク２１２、ＣＤＲＯＭ２１０をアクセスするものである。

【００７５】

また、ここには、ハードディスクコントローラ２２３によりアクセスされるハードディスク２１１、マウスコントローラ２２６により制御されるマウス１０４、キーボードコントローラ２２７により制御されるキーボード１０３、およびディスプレイコントローラ２２８により制御されるＣＲＴディスプレイ１０２も示されている。

【００７６】

前述したように、ＣＤＲＯＭ２１０には機構制御プログラム開発支援プログラムが記憶されており、ＣＤＲＯＭドライバ２２５により、そのＣＤＲＯＭ２１０からその機構制御プログラム開発支援プログラムが読み込まれ、バス２２０を経由し、ハードディスクコントローラ２２３によりハードディスク２１１内に格納される。実際の実行にあたっては、そのハードディスク２１１内のプログラムは

ＲＡＭ２２２上にロードされ、ＣＰＵ２２１により実行される。

【００７７】

図３は、プログラム記憶媒体内に記憶されたプログラムの構成を示す図である。

【００７８】

この図３に示すプログラム記憶媒体３００は、機構制御プログラム開発支援プログラムが記憶されてなるＣＤＲＯＭ２１０、そのプログラムがインストールされた状態のハードディスク２１１、およびそのプログラムがフロッピィディスクにダウンロードされたときの、そのダウンロードされたプログラムを記憶した状態にあるフロッピィディスク２１２等を代表的に示したものである。

【００７９】

この図３に示すプログラム記憶媒体３００には機構制御プログラム開発支援プログラム３１０が記憶されている。この機構制御プログラム開発支援プログラムは、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレーションプログラム３１１と、その三次元機構モデルを制御する制御プログラムに付加される同期化プログラム３１２とから構成されている。この同期化プログラムは、制御プログラムに付加されて、その制御プログラムを、三次元機構モデルシミュレーションプログラム３１１の実行による三次元機構モデルの動作との同期をとりながら実行させるプログラムである。

【００８０】

機構制御プログラム開発支援プログラム３１０を構成する、三次元機構モデルシミュレーションプログラム３１１と同期化プログラム３１２は、図１に示すコンピュータシステム内で実行される際、相互に以下のような通信を行なうことにより、三次元機構モデルの動作と制御プログラム実行との同期がとられる。すなわち、三次元機構モデルシミュレーションプログラム３１１が、制御プログラムの実行時間を同期化プログラム３１２に通知する。同期化プログラム３１２は、その通知を受けて、その通知を受けた実行時間分だけ制御プログラムの実行を許容し、制御プログラムがその実行時間だけ実行されると制御プログラムの実行をその段階で中断させ、制御プログラムのその実行時間分だけの実行が終了したことを三次元機構モデルシミュレーションプログラム３１１に通知する。三次元機構モデルシミュレーションプログラム３１１は、その同期化プログラム３１２からの通知を受けて、三次元機構モデルを、動作プログラムの、その実行時間分の制御に応じた動作量だけ動作させて、制御プログラムの次の実行時間を決定し、その決定した実行時間を同期化プログラム３１２に通知する。これを繰り返すことにより、相互に同期がとられた形態で、動作プログラムが実行され、その動作プログラムの実行による制御に従って三次元機構モデルが動作する。

【００８１】

図４は、本発明の機構制御プログラム開発支援システムの一例が構築されたコンピュータネットワークの一例を示す図である。

【００８２】

図４には、２台のコンピュータシステム４００，５００が通信ネットワーク６００を介して接続されている様子が示されており、ここでは、コンピュータシステム４００が本発明にいう三次元機構モデルシミュレータを構成し、コンピュータシステム５００が本発明にいう制御プログラムプロセッサを構成している。

【００８３】

各コンピュータシステム４００，５００は、ＣＰＵ、ＲＡＭ、ハードディスク、通信用モデム等を内蔵した本体部４０１，５０１、本体部４０１，５０１からの指示により表示画面４０２ａ，５０２ａに画面表示を行うＣＲＴディスプレイ４０２，５０２、ユーザの指示や文字情報を入力するためのキーボード４０３，５０３、表示画面４０２ａ，５０２ａ上の任意の位置を指定することによりその位置に表示されていたアイコン等に応じた指示を入力するマウス４０４，５０４

を備えている。

【００８４】

本体部４０１，５０１には、外観上、フロッピィディスクやＣＤＲＯＭが取り出し自在に装填されるフロッピィディスク装填口４０１ａ，５０１ａやＣＤＲＯＭ装填口４０１ｂ，５０１ｂを有しており、その内部には、装填されたフロッピィディスク、ＣＤＲＯＭをドライブするフロッピィディスクドライバ、ＣＤＲＯＭドライバも内蔵されている。

【００８５】

ここでは、機構制御プログラム開発支援プログラムのうちの三次元機構モデルシミュレーションプログラム３１１（図３参照）は、コンピュータシステム４００にインストールされ、同期化プログラム３１２は、コンピュータシステム５００にインストールされる。コンピュータシステム４００内には三次元機構モデルが構築されており、コンピュータシステム４００内にインストールされた三次元機構モデルシミュレーションプログラム３１１は、そのコンピュータシステム４００内に構築された三次元機構モデルを動作させる。一方、コンピュータシステム５００には、コンピュータシステム４００内に構築された三次元機構モデルを実際に製造したときの製品を制御するための制御プログラムも格納されており、このコンピュータシステム５００にインストールされた同期化プログラムは制御プログラムと統合され、コンピュータシステム５００内で動作する三次元機構モデルの動作を制御する。三次元機構モデルシミュレーションプログラム３１１と同期化プログラム３１２との間の通信は、通信ネットワーク６００を経由して行なわれる。

【００８６】

本発明の機構制御プログラム開発支援システムには、この図４に示すように、三次元機構モデルシミュレータと制御プログラムプロセッサがそれぞれ別々の装置（ここではコンピュータシステム４００とコンピュータシステム５００）に組み込まれ、それらの間で通信を行なうように構成された形態も含まれる。

【００８７】

また、本発明の機構制御プログラム開発支援システムには、以下のような形態も存在する。すなわち電源や水晶を繋げたマイコンチップに制御プログラムをローディングし、例えば図４におけるコンピュータシステム４００にＩ／Ｏボードを挿入してマイコンチップと接続し、マイコンチップから出力された、本来はモータドライバに伝達される電気信号をＩ／Ｏボードを介してコンピュータシステム４００に入力し、コンピュータシステム４００では三次元機構モデルを動作させて本来センサから出力される信号と同じ信号をＩ／Ｏボードを介して出力し、その信号をマイコンチップで受信する。

【００８８】

このようにして、三次元機構モデルはコンピュータシステム４００内で動作させ、制御プログラムは実際の使用形態に近いマイコンチップで動作させるという環境を実現してもよい。

【００８９】

図５は、図４に外観を示すコンピュータネットワークを構成するコンピュータシステムのハードウェア構成図である。

【００９０】

図４に示すコンピュータシステム４００，５００は、本実施形態ではハードウェア構成は同一であり、ここでは、コンピュータシステム４００を例に挙げて説明する。

【００９１】

ここには、中央演算処理装置（ＣＰＵ）４２１、ＲＡＭ４２２、ハードディスクコントローラ４２３、フロッピィディスクドライバ４２４、ＣＤＲＯＭドライバ４２５、マウスコントローラ４２６、キーボードコントローラ４２７、およびディスプレイコントローラ４２８が示されており、それらはバス４２０で相互に

接続されている。

【００９２】

またここには、ハードディスク４１１，フロッピィディスク４１２，ＣＤＲＯＭ４１０，マウス４０４，キーボード４０３，ＣＲＴディスプレイ４０２も示されている。これらの各要素は、図２に示したハードウェア構成図における対応する各構成要素とそれぞれ同じ作用を成すものであり、詳細説明は省略する。

【００９３】

図２には示されていない要素として、この図５には、モデム４２９が示されている。このモデム４２９は、図４に示す通信ネットワーク６００に接続され、２台のコンピュータシステム４００，５００間での通信を担当するものである。

【００９４】

以上では、図１および図２と、図３および図４との２種類のハードウェア構成を示したが、以下ではそれらのコンピュータシステムあるいはコンピュータネットワーク内に構築された機構制御プログラム開発支援システムの具体的な内容について説明する。

【００９５】

図６は、機構制御プログラム開発支援システムの概念図である。

【００９６】

三次元機構モデルシミュレータ（「３Ｄモデルシミュレータ」と略記する）内には三次元機構モデル（「３Ｄモデル」と略記する）が構築されており、一方、制御プログラムプロセッサ内には、その３Ｄモデルを制御する制御プログラムがインストールされている。

【００９７】

この制御プログラムプロセッサ内の制御プログラムは、３Ｄモデルに相当する製品を実際に製作したときの、その製品を制御する制御プログラムに、３Ｄモデルシミュレータ内で動作する３Ｄモデルの動作との同期をとるための同期化プログラムとが結合したものである。

また３Ｄモデルは、複数のリンク（製品を構成する個々の部品）の三次元形状とそれらのリンクの姿勢の関係を定義した三次元機構データによって定義される、製品の三次元モデルであり、リンクの姿勢を変更することによってその３Ｄモデルが動作することになる。

【００９８】

制御プログラムプロセッサから３Ｄモデルシミュレータにはモータ等のアクチュエータを動作させるためのアクチュエータ指令信号が送られ、３Ｄモデルシミュレータから制御プログラムプロセッサには、センサのオン、オフ等をあらわすセンサ信号が送られる。また、これら制御プログラムプロセッサと３Ｄモデルシミュレータとの間では同期信号が送受信される。

【００９９】

図７は、センサやモータの定義時の処理を示す３Ｄモデルシミュレータの模式図、図８は、シミュレーション時の３Ｄモデルシミュレータの処理を示す模式図である。

【０１００】

３Ｄモデルは、三次元ＣＡＤ等を用いて構築され、その構築された３Ｄモデルに対し、ここでは、図７に示すように、ユーザによりキーボード等の入力装置が操作されてモータやセンサが定義される。そのような定義が行なわれた後の実際のシミュレーションにあたっては、制御プログラムプロセッサから送られてきたアクチュエータ指令信号（ここではモータ回転指令信号）に対応し、モータとして定義されたリンク（以下、モータとして定義されたリンクを「モータリンク」と称する）の姿勢を変化させ、さらに機構データに基づき、モータリンクに連結された他のリンクについて、モータリンクの姿勢変化に応じた量だけ動作させ、その結果として、センサとして定義されたリンク（以下、センサとして定義されたリンクを「センサリンク」と称する）の姿勢にも影響が出る。そこでセンサリ

ンクの姿勢に応じセンサ信号を発生させ、そのセンサ信号は制御プログラムプロセッサに送信される。

【０１０１】

図９は、シミュレーション時における３Ｄモデルシミュレータと制御プログラムの動作タイミングチャートを示す図である。

【０１０２】

図中のハッチング部分は、シミュレータ、制御プログラムがそれぞれ動作していることを示している。図示のように、シミュレータと制御プログラムは動作、停止のサイクルを交互に繰り返している。各サイクルは次のような手順で動作する。ただし、これらシミュレータと制御プログラムとの間の同期に関する詳細な説明は後にゆずる。

【０１０３】

（１）シミュレータは、入力アクチュエータ信号を基に、シミュレーション時間Δｔ分のシミュレーションを行ない、センサ信号を出力する。

【０１０４】

（２）シミュレータは、現在の３Ｄモデルの干渉可能性から、次のループでのシミュレーション時間Δｔを決定し、その時間Δｔを制御プログラムに受け渡すとともに、動作開始指令（Ｒｅａｄｙ　Ｆｌａｇ）を制御プログラムに受け渡す。

【０１０５】

（３）制御プログラムは時間Δｔを受け取るとともに動作開始指令（Ｒｅａｄｙ　Ｆｌａｇ）を受け取って、その時間Δｔ分の計算（制御）を行なった後、シミュレータ側に動作終了信号（Ｌｏｏｐ　Ｆｌａｇ）を送り、次の動作開始指令（Ｒｅａｄｙ　Ｆｌａｇ）を受け取るまでの間、動作を停止する。

【０１０６】

（４）シミュレータは制御プログラムの動作終了信号（Ｌｏｏｐ　Ｆｌａｇ）を受け取って時間Δｔ分のシミュレーションを開始する。

【０１０７】

シミュレータと制御プログラムは、以上を繰り返すことにより交互に動作する。

【０１０８】

図１０は、シミュレーション時のシミュレータのフローを示すフローチャートである。

【０１０９】

シミュレーションが開始されると、先ず初期設定が行なわれて（ステップＳ１０１）、３Ｄモデルの各リンクの姿勢が初期化される。次いで、シミュレーションを行なう時間間隔（シミュレーション間隔）Δｔが制御プログラム側に送信され（ステップＳ１０２）、制御プログラム側からシミュレーション間隔Δｔを受信した旨をあらわす信号を受け取ると（ステップＳ１０３）、制御プログラムに制御プログラム計算開始命令を送信する（ステップＳ１０４）。制御プログラムによりシミュレーション間隔Δｔ分の計算が行なわれてその終了が通知されると（ステップＳ１０５）、ここでシミュレートしている３Ｄモデルを構成する全てのモータおよび全てのセンサを今回のΔｔ分の処理に関し未処理である旨初期設定される（ステップＳ１０６）。

【０１１０】

次いでステップＳ２０１では、未処理のモータが存在しているか否か判定され、未処理のモータ１つずつについてステップＳ２０２～Ｓ２０７の処理が行なわれる。すなわち、制御プログラム側から送られてきた、今処理しようとしているモータに関するアクチュエータ信号に応じてそのモータの回転速度が決定され（ステップＳ２０２）、その回転速度からΔｔ間に変位するモータの姿勢が決定される（ステップＳ２０３）。次いで機構関係定義によりそのモータに関連づけられた他の部品の姿勢が変更され（ステップＳ２０４）、部品間の最短距離ｄが測

定される（ステップＳ２０５）。ステップＳ２０６、Ｓ２０７では、各モータの回転により測定される各最短距離ｄの中の最短距離Ｄが求められる。このようにして、全てのモータに関し、Δｔの間の姿勢変更が終了すると、姿勢変更後の３Ｄモデルが画面に表示される（ステップＳ２０８）。

【０１１１】

次いで、今度は未処理のセンサが存在しているか否か判定され（ステップＳ３０１）、未処理のセンサ１つずつについてセンサリンクの姿勢に応じたセンサ信号のオン、オフが検出される（ステップＳ３０２）。全てのセンサについての処理が終了すると、制御プログラムに向けてそれら全てのセンサに関するセンサ信号が出力され（ステップＳ３０３）、さらに部品間最短距離Ｄに応じたサンプリング間隔Δｔが決定され（ステップＳ４０１）、ステップＳ１０２に戻ってその決定された間隔Δｔが制御プログラムに送信される。

【０１１２】

シミュレータでは、以上の処理が繰り返し実行され、制御プログラムとの同期をとりつつ、３Ｄモデルの動作シミュレーションが行なわれる。

【０１１３】

図１１は、図１０に示すシミュレータの処理フローの一部分（図１０に一点鎖線で囲った部分）に代えて採用することのできる部分フローを示すフローチャートである。

【０１１４】

この図１１には、ステップＳ２０２とステップＳ２０３との間にステップＳ２１０が挿入されている。このステップＳ２１０では、今処理を行なっているモータの回転速度がゼロ（ω＝０）か否かが判定され、今回の間隔Δｔの間にそのモータは回転しない（ω＝０）のときはそのモータの動作に関しては部品間の距離ｄの測定は行なわずにステップＳ２１１に進んでｄ＝∞とし、ステップＳ２０６の判定でその距離ｄ＝∞が距離Ｄを求める演算から除外されるようにしている。

【０１１５】

部品間の距離を求めるにはかなりの計算量を要し、したがってモータが回転しないときはそのモータに関する距離の測定を行なわないようにすることにより、計算量を減らし、シミュレーションのスピードアップを図ることができる。

【０１１６】

図１２は、モータとして定義されるリンクの選び方を示す模式図である。

【０１１７】

この図１２において、本来のモータは図中のリンクＡであるが、リンクＡ自体は土台に固定されているため、シミュレーション上は、その本来のモータによって一番最初に駆動されて３Ｄモデル上姿勢が変更されるリンクＢがモータとして定義される。モータをこのように定義することでシミュレーションが容易となる。

【０１１８】

図１３，図１４は、センサとして定義されるリンクの選び方を示す模式図である。

【０１１９】

これら図１３，図１４において、本来のセンサはリンクＡであるが、ここでは、その本来のセンサによって検出されるリンクＢをセンサとして定義し、そのリンクＢの姿勢に応じて、図１３（Ａ），図１４（Ａ）の場合はオフ、図１３（Ｂ），図１４（Ｂ）の場合はオンと定義される。このように、センサの場合も、モータの場合と同様、本来のセンサではなくそのセンサにより検出されるリンクをセンサとして定義し、そのセンサリンクの姿勢に応じてオン、オフを定義することにより、シミュレーションが容易となる。

【０１２０】

図１５は、モータの種類を示す模式図である。

【０１２１】

本実施形態では、モータを定義するにあたり「無段変速」と「有段変速」との二種類の定義方法が用意されている。

【０１２２】

無段変速は、速度指令値と目標速度との間の比例定数を定義する定義方法である。この無段変速はさらに、正負に関係なく比例させる方法（例えば比例定数が１００のとき、二進数１０１（１０進数で－３）が－３００ｒｐｍ（マイナス符号は逆転方向への回転をあらわす）に変換される）と、二進数の最上位ビットを符号とみなす方法（例えば二進数で１０１のとき、最上位ビット‘１’とその他のビット‘０１’とに分け、最上位ビット‘１’はマイナス符号（‘０’はプラス符号）、残りの‘０１’を数値（１０進数で１）とし、それらを合わせた－１に比例定数１００を掛けて－１００ｒｐｍに変換する方法）との２種類が用意されている。

【０１２３】

有段変速は、速度指令値と目標速度との対応表が定義される。例えば図１５では、１０１に対し－２００ｒｐｍが定義されており、速度指令値として‘１０１’が入力されると目標速度として－２００ｒｐｍが出力される。

【０１２４】

本実施形態ではこのような複数種類の定義方法を用意しておくことにより、モータの定義の自由度を向上させている。

【０１２５】

図１６は、モータを一次遅れ系とみなしたときの、モータの回転速度ωの初期変化を示す図である。

【０１２６】

ここでは、モータの回転速度変化を図１６に示すような一次遅れ系としてとらえて、モータの負荷による遅れやその影響を取り入れたシミュレーションを行なう。モータを一次遅れ系とみなすと、ユーザは、整定時間４Ｔを入力するだけでそのモータの特性を指定することができ、入力作業の繁雑さが低減する。

【０１２７】

以下に、モータを一次遅れ系とみなしたときの、シミュレーション間隔Δｔの間のモータの回転量Δθの算出式を導出する。

【０１２８】

時刻ｔにおけるモータ速度ωは、

【０１２９】

【数１】

$$\omega = \omega_o + (\omega_i - \omega_o)(1 - e^{-\frac{1}{T}}) \qquad \cdots\cdots (1)$$

【０１３０】

によりあらわされる。ここでω_iは目標速度、ω_oは目標速度指令前のモータ速度、Ｔはモータの性質を表す、整定時間４Ｔに比例した定数である。式（１）より、時刻ｔにおけるモータ速度をω_{k-1}とすると、

【０１３１】

【数２】

$$\omega_{k-1} = \omega_o + (\omega_i - \omega_o)(1 - e^{-\frac{1}{T}}) \qquad \cdots\cdots (2)$$

【０１３２】
同様にｔよりΔｔ後の時刻ｔ＋Δｔのモータ速度ω_kは
【０１３３】
【数３】

$$\omega_k = \omega_o + (\omega_i - \omega_o)(1 - e^{-\frac{t+\Delta t}{T}}) \qquad \cdots\cdots (3)$$

【０１３４】
となる。
式（３）から式（２）を引くと
【０１３５】
【数４】

$$\omega_k - \omega_{k-1} = (\omega_i - \omega_o)(1 - e^{-\frac{t+\Delta t}{T}}) - (\omega_i - \omega_o)(1 - e^{-\frac{1}{T}})$$

$$= (\omega_i - \omega_o)\left\{ (1 - e^{-\frac{t+\Delta t}{T}}) - (1 - e^{-\frac{1}{T}}) \right\}$$

$$= (\omega_i - \omega_o)(e^{-\frac{1}{T}} - e^{-\frac{t+\Delta t}{T}})$$

$$= (\omega_i - \omega_o)e^{-\frac{1}{T}}(1 - e^{-\frac{\Delta t}{T}}) \qquad \cdots\cdots (4)$$

$$\omega_k = \omega_{k-1} + (\omega_i - \omega_o)e^{-\frac{1}{T}}(1 - e^{-\frac{-\Delta t}{T}}) \qquad \cdots\cdots (5)$$

【０１３６】
となる。
さて、ここで（２）式を以下のように書き換える。
【０１３７】
【数５】

$$\omega_{k-1} - \omega_o = (\omega_i - \omega_o)(1 - e^{-\frac{1}{T}})$$

$$= (\omega_i - \omega_o) - (\omega_i - \omega_o)e^{-\frac{1}{T}}$$

$$(\omega_i - \omega_o)e^{-\frac{1}{T}} = \omega_i - \omega_o - \omega_{k-1} + \omega_o$$

$$= \omega_i - \omega_{k-1} \qquad \cdots\cdots (6)$$

【０１３８】

式（６）を（５）に代入することにより以下の式が得られる。

【０１３９】

【数６】

$$\omega_k = \omega_{k-1} + (\omega_i - \omega_{k-1})(1 - e^{-\frac{\Delta t}{T}}) \qquad \cdots\cdots (7)$$

【０１４０】

Δｔの間にモータが回転する量は、式（７）をΔｔについて積分すれば良いから

【０１４１】

【数７】

$$\Delta\theta = \int_0^{\Delta t} \omega_{k-1} + (\omega_i - \omega_{k-1})(1 - e^{-\frac{1}{T}})dt$$

$$= \omega_{k-1}\Delta t + (\omega_i - \omega_{k-1})\{\Delta t - T(1 - e^{-\frac{\Delta t}{T}})\} \qquad \cdots\cdots (8)$$

【０１４２】

となる。モータを一次遅れ系とみなしたとき、この（８）式に基づいて、目標速度ω_i、シミュレーション時間Δｔ、そのシミュレーション時間Δｔだけ前の時刻におけるモータ速度ω_{k-1}、および整定時間４Ｔによって、そのシミュレーション時間Δｔの間のモータの回転量Δθが求められる。

【０１４３】

ただし、モータは、常に上記（８）式のように動作する必要はなく、例えばモータに指令される目標速度ω_iが変化してから整定時間４Ｔが経過した時点で速度をω_iに固定してもよい。

【０１４４】

図１７は、シミュレータにおける、一次遅れを考慮したモータ駆動の処理を示すフローチャートである。

【０１４５】

部品間距離よりシミュレーション間隔Δｔが決定され（ステップＳ５０１）、その後さらに制御プログラムからアクチュエータ信号が入力されると（ステップＳ５０２）、図１５に示すモータの定義に従って比例定数あるいは対応表が参照されて目標速度ω_iが決定される（ステップＳ５０３）。

【０１４６】

ステップＳ５０４では、目標速度ω_iが変更されたか否かが判定され、今回目標速度ω_iが変更されていたときはステップＳ５０５に進み、目標速度ω_iが変更された時点からの経過時間を示すt_{total}が$t_{total}=0$に初期化され、次回のステップＳ５０４での判定のためにω_{iold}にω_iが格納される（ステップＳ５０６）。

【０１４７】

ステップＳ５０４で$\omega_i=\omega_{iold}$であると判定された場合は直接に、$\omega_i\neq\omega_{iold}$であったときはステップＳ５０５，Ｓ５０６を経由した後に、ステップＳ５０７に進み、目標速度ω_iが変更された時点からの経過時間t_{total}が整定時間４Ｔに達したか否かが判定される。t_{total}が未だ４Ｔに達していないときは、ステップＳ５０８に進み、上述の式（７）に従ってモータの回転速度ωが求められ、さらにステップＳ５０９において、式（８）に従ってモータの回転量Δθが求められる。

【０１４８】

一方、ステップＳ５０７において$t_{total}\geqq 4T$であると判定されたときはステップＳ５１０に進み、モータの回転速度ωが目標速度ω_iと等しい速度に設定され、さらにステップＳ５１１において、モータの回転量Δθが、目標速度ω_iで時間Δｔだけ回転したときの回転量に設定される。

【０１４９】

ステップＳ５０８，Ｓ５０９を経由した場合であっても、あるいはステップＳ５１０，Ｓ５１１を経由した場合であっても、ステップＳ５１２においてt_{total}がΔｔだけ増分され、モータのトータルの回転量θがΔθだけ増分される。

【０１５０】

この図１７に示す処理が制御プログラムと同期をとりながら繰り返し実行されることにより、図１６に示す一次遅れ系とみなしたときの速度変化に従ってモータが回転する。

【０１５１】

図１８～図２０は、干渉チェックを用いたセンサの定義方法の説明図である。

【０１５２】

図１８には、接触センサの姿勢が複数のモータの姿勢の組合せにより決定される例が示されており、この場合、図１３、図１４を参照して説明した、実際のセンサにより検出されるリンクをセンサとみなす方法を採用すると、複数のモータそれぞれの姿勢の組合せでセンサのオン，オフを定義することになり極めて複雑な定義が必要となる。そこでここでは、センサと土台との干渉をチェックし、それらの間の干渉の有無によってセンサをオンあるいはオフとする。こうすることにより、ユーザに複雑な定義を要求することなく、センサのオン、オフを決定することができる。

【０１５３】

図１９は、スリットの移動により光量センサのオン、オフが多数回繰り返される例が示されており、この場合、図２０に示すように光束を１つの仮想リンクと見なし、その仮想リンクとスリットとの干渉の有無によってセンサのオン、オフを判定する。

【０１５４】

この場合、仮想リンクとの干渉をチェックする相手のリンクは、図１９に示すように相手のリンクがあらかじめわかっているときはそのリンクに限定してもよく、あるいは相手のリンクが不明である場合であっても、本来のセンサ自体をあらわすリンクは、その仮想リンクと常に干渉（接触）しているため、干渉をチェ

ックする相手のリンクからは除外される。

【０１５５】

この仮想リンクによるセンサ出力決定方法は、ここに示した光電センサなどに限らず、例えばマイクロスイッチの接触部分を仮想リンクとして定義するなど、接触式の位置検出センサにも適用することができる。

【０１５６】

図１３、図１４を参照して説明したセンサの定義方法は、ポテンションメータ、エンコーダ等の角度を検出するセンサなど、一自由度の変位を検出するセンサを表現するのに有効な方法であるが、ここに示したような、複数のモータの姿勢の組合せにより出力が決定されるように配置されたセンサ（図１８）など多自由度の組合せの結果として姿勢が決定されるセンサや、スリットカウンタ（図１９）などオン、オフの変化が多いセンサにはその定義方法を採用するとユーザに複雑な定義を要求することになるため、ここに説明した、リンクどうしの干渉の有無によりセンサオン、オフを判定する方法を採用することが好ましい。

【０１５７】

図２１は、シミュレータと制御プログラムとの間で同期をとるために送受信される信号のタイミングチャート、図２２は、その同期を実現するための３Ｄモデルの動作との同期をとるための同期化プログラムを示すフローチャート、図２３は、その同期を実現するための、シミュレータ側の処理を示すフローチャートである。ここで、同期化プログラムは、３Ｄモデルに対応する製品を製造したときのその製品の動作を制御する制御プログラムに追加されるプログラムである。

【０１５８】

図２１に示す、ループフラグ（Ｌｏｏｐ　Ｆｌａｇ）、レディフラグ（Ｒｅａｄｙ　Ｆｌａｇ）、およびタイムスケールは、それぞれ以下の意味を持つ信号である。

【０１５９】

（１）Ｌｏｏｐ　Ｆｌａｇ

制御プログラムからシミュレータへ受け渡す信号であって、制御プログラムが現在動作中であるか否かを表わしている。

【０１６０】

（２）Ｒｅａｄｙ　Ｆｌａｇ

シミュレータから制御プログラムへ受け渡す信号であって、シミュレータによる１つのシミュレータ間隔Δｔ分のシミュレーションが終了したことを表わしている。

【０１６１】

（３）タイムスケール

シミュレータから制御プログラムへ受け渡す信号であって、シミュレータが部品間最短距離に基づいて決定したシミュレーション間隔Δｔをあらわしている。

【０１６２】

一般に、制御プログラムは、メインループやタイマ割り込みルーチンなど、一定周期で動作する部分が存在する。同期処理のために制御プログラムに追加される同期化プログラムは最小限の規模のものであることが好ましく、ここに示す例では、タイムスケールとして、５ｍｓｅｃ，２ｓｅｃなどの具体的な時間ではなく、この一定周期で動作するルーチンを通過する回数ｎで指定している。

【０１６３】

図２２に示すルーチンはそれをあらわしており、ステップＳ６０２でｎ＝０か否かを判定し、ｎ＝０でないときはステップＳ６０３に進んでｎを１だけデクリメントし、本来の制御プログラムの処理の実行に移っている。ステップＳ６０１においてｎ＝０と判定されたときのみ、ステップＳ６０３～Ｓ６０７の同期のための処理が実行されるように構成されている。

【０１６４】

ここでは、図２１～図２３を説明するにあたり、シミュレータでは現在シミュ

レーションが行なわれており、したがってＲｅａｄｙ　Ｆｌａｇは‘Ｌｏｗ’レベルにあり、制御プログラム側では図２２のステップＳ６０４にとどまってＲｅａｄｙ　Ｆｌａｇが‘Ｈｉｇｈ’に変化するを待っている段階にあるとする。このとき、シミュレータにおける同期処理をあらわす図２３のルーチンでは、ステップＳ７０３のシミュレーションを行なっている途中にある。

【０１６５】

ここで、シミュレーションが終了すると、シミュレータは図２３のステップＳ７０４に移り、次のシミュレーション間隔（タイムスケール）Δｔ（ただし上述したように制御プログラムの動作回数ｎであらわされている）が決定され、ステップＳ７０５においてそのタイムスケールΔｔ（ｎ）が制御プログラムに送信され、さらにステップＳ７０６に進んでＲｅａｄｙ　Ｆｌａｇが‘Ｈｉｇｈ’レベルに変更される。

【０１６６】

すると、制御プログラム側では、図２２に示すステップＳ６０４を抜け出してステップＳ６０５に進みタイムスケールｎが読み込まれ、さらにステップＳ６０６に進んでＬｏｏｐ　Ｆｌａｇが‘Ｈｉｇｈ’に変更される。その後ステップＳ６０７において、シミュレータ側から送信されてきているＲｅａｄｙ　Ｆｌａｇが‘Ｌｏｗ’に変化するのを待って本来の制御処理に進む。前述したように、この制御プログラムは一定周期で繰り返し実行されており、ステップＳ６０１ではｎ＝０か否かが判定され、ｎ≠０のときはステップＳ６０２に進んでｎが１だけデクリメントされる。これを繰り返し、ステップＳ６０１でｎ＝０と判定されるとステップＳ６０３に進み、Ｌｏｏｐ　Ｆｌａｇが‘Ｌｏｗ’に変更され、その後、ステップＳ６０４において、シミュレータ側から送られてくるＲｅａｄｙ　Ｆｌａｇが‘Ｈｉｇｈ’に変化するのを待つ状態となる。

【０１６７】

一方、シミュレータ側では、図２３のステップＳ７０６においてＲｅａｄｙ　Ｆｌａｇが‘Ｈｉｇｈ’に変更された後、ステップＳ７０７において、制御プログラム側から送られてくるＬｏｏｐ　Ｆｌａｇが‘Ｈｉｇｈ’に変化するのを待ち、Ｌｏｏｐ　Ｆｌａｇが‘Ｈｉｇｈ’に変化したことを受けてステップＳ７０８に進み、Ｒｅａｄｙ　Ｆｌａｇを‘Ｌｏｗ’に変化させ、ステップＳ７０２に進んで、Ｌｏｏｐ　Ｆｌａｇが‘Ｌｏｗ’に変化するのを待つ。Ｌｏｏｐ　Ｆｌａｇが‘Ｌｏｗ’に変化すると、ステップＳ７０３に進み、シミュレーション間隔Δｔ分のシミュレーションが行なわれる。

【０１６８】

以上の処理が繰り返され、シミュレータと制御プログラムでは、同期がとられながらそれぞれの処理が実行される。

【０１６９】

次に、リンク（部品）どうしの干渉可能性の評価方法について説明する。

【０１７０】

図２４は、干渉可能性の評価方法の説明図である。ここには、リンクＡが速度ｖでリンクＢに向かって動いている例が示されている。

【０１７１】

図２４（Ａ）は干渉する可能性のある２つのリンク（リンクＡとリンクＢ）が大きく離れている状態を示しており、この場合は干渉可能性が小さいものと判定されてシミュレーション間隔Δｔとし比較的大きな値が設定され、リンクＡは一回のシミュレーションでｖΔｔだけ進む。この場合、シミュレーションを高速に行なうことができる。

【０１７２】

また、図２４（Ｂ）は、リンクＡとリンクＢとが近づき、それらの間の距離が小さい状態をあらわしている。この場合、干渉可能性が大きいものと判定され、シミュレーション間隔Δｔとして比較的小さな値が設定される。このときは、リンクＡは、Δｔが小さいため一回のシミュレーションで進む距離ｖΔｔが小さく

、高精度なシミュレーションを行なうことができる。

【０１７３】

このように、本実施形態では、部品間の距離に応じてシミュレーション間隔Δｔが変更され、高精度なシミュレーションと高速なシミュレーションとの両立、調和が図られている。

図２５は、部品間の距離ｄとシミュレーション間隔Δｔとの関係の各種の例を示す図である。

【０１７４】

これらの例に示すように、距離ｄが小さくてもシミュレーションがあまりにも低速化されないよう、かつ距離ｄが大きくてもシミュレーションがあまりにも高速化されないようにその調整が図られている。

【０１７５】

次に、干渉可能性の評価のために部品をグループ分けする方法について説明する。

【０１７６】

図２６は、部品のグループ分けの説明図である。

【０１７７】

ここにはリンクl_0～リンクl_5の５つのリンクが示されており、モータリンクとして定義されたリンクl_3が動く場合を考える。この場合、モータリンクl_3の動きによって姿勢に影響を受けるリンクはリンクl_3～l_5であり、これらのリンクl_3～l_5は、モータリンクl_3の動きに対して１つの剛体と考えることができる。また、モータリンクl_3の動きによっては姿勢に影響を受けないリンクl_0～l_2も同様に、これを合わせたものを１つの剛体と考えることができる。そこでモータリンクl_3を動作させる場合には、そのモータリンクl_3を動かしたことにより姿勢に影響を受けるリンクのグループと影響を受けないリンクのグループとの２つのグループに分けてそれら２つのグループそれぞれを剛体とみなし、それら２つの剛体の間の最短距離を求めることにより、その求めた距離を次のシミュレーション間隔Δｔを決定するための最短距離の候補の１つとすることができる。ただし、この図２６におけるリンクl_3とリンクl_2、例えばモータの軸と歯車等は、常に接触しているにも拘らず異なるグループに分けられてしまい、このままでは上記２つの剛体は常に干渉していることになってしまう。そこでここでは、上記のようにして２つのグループに分けた後、一度部品間距離を求め、既に接触、干渉しているリンクのうち、今注目しているモータリンクを動かしたときに動かない方のグループに属するリンク（図２６に示す例の場合リンクl_2）をそのグループから除外した上で２つのグループそれぞれを剛体とみなし、それら２つの剛体の間の距離が求められる。

【０１７８】

図２７は、１つの３Ｄモデル内に複数のモータリンクが定義されている場合のグループ分けの説明図である。

【０１７９】

図２７（Ａ）は、３Ｄモデルをあらわしており、ここには、モータ１～モータ３の３つのモータリンクが定義されている。このとき、図２６を参照して説明したグループ分けを、モータ１のみを動かしたとき（図２７（Ｂ））、モータ２のみを動かしたとき（図２７（Ｃ））、モータ３のみを動かしたとき（図２７（Ｄ））のそれぞれについて行なう。こうしておいて、シミュレーション実行時には、対象としているモータリンクを動かしたときは、そのモータリンクに関しグループ分けしたグループ間の最短距離が求められ、それら複数のモータリンクの動きにより求められた複数の‘最短距離’の中からさらに最短の距離が求められる（図１０ステップＳ２０１～Ｓ２０７参照）。

【０１８０】

図２８は、グループ分けした部品の再グループ化の説明図である。

【０１８１】

図２７を参照して説明した手法によると、各モータリンクに関し、その３Ｄモデルを構成するほぼ全ての部品がどちらかのグループ（剛体）に属する結果となり、この場合、各剛体を定義するためのデータ量が厖大なものとなってしまい極めて大きなメモリ容量を必要とする可能性がある。そこで、図２８に示すように、各モータリンクの動きによって動かない方のグループ（図２８に示す例ではグループＤ，Ｅ，Ｆ）に属する部品群のうち、共通部分を抜き出して１つの剛体とするとともに、各グループＤ，Ｅ，Ｆからその共通部分を除いた部分をそれぞれ１つの剛体とする。こうしておいて、距離の測定の際は、共通部分からなる剛体とその共通部分を除いた部分の剛体とを合体させて、相手のグループ（グループＤ，Ｅ，Ｆに対応してそれぞれグループＡ，Ｂ，Ｃ）との間の距離を求めるようにしてもよい。こうすることにより、共通部分を各グループに配置しておく場合（図２７の場合）と比べ、これらのグループを定義するデータを格納しておくためのメモリの容量を減らすことができる。

図２９は、モータリンクの動きによって姿勢に影響を受けるリンクのグループとそのモータリンクの動きによっては姿勢に影響を受けないグループとに分けるルーチンを示すフローチャート、図３０は、そのサブルーチンである検索ルーチンのフローチャートである。

【０１８２】

ここでは、モータリンクの動きによって姿勢に影響を受けるリンクがリストＡにリストアップされ、それ以外のリンクがリストＢにリストアップされる。尚、リストＡおよびリストＢは、ここで説明するフローの実行に先立って白紙状態にクリアされているものとする。

【０１８３】

ここでは、先ずリンクを指し示すポインタＰによりモータリンクが指し示され（図２９、ステップＳ８０１）、そのポインタＰで指し示されるリンクがリストＡに追加される。

【０１８４】

次にポインタＰで指し示されているリンクにチャイルド（ｃｈｉｌｄ）のリンクが存在しているか否かが判定される。ｃｈｉｌｄとは、例えば図２６に示すリンク機構においてポインタがモータリンクl_3を指し示していたとき、そのモータリンクl_3が動くことによって一緒に動くリンクの１つ（ここではリンクl_4）をいう。リンクl_4が仮にモータリンクであったとしても今着目しているモータリンクはリンクl_3であり、リンクl_4はリンクl_3のｃｈｉｌｄとなる。尚、ポインタＰが指し示すリンクに同格の複数のｃｈｉｌｄが考えられるときであってもｃｈｉｌｄはそのうちの１つに限定され、ｃｈｉｌｄと考えられる他のリンクは、その１つの限定されたｃｈｉｌｄのブラザ（ｂｒｏｔｈｅｒ）として定義される。

【０１８５】

ポインタＰが指し示すリンクにｃｈｉｌｄがあったときは、ステップＳ８０４に進み、そのｃｈｉｌｄのリンクを指し示すようにポインタＰを移し、図３０に示す検索ルーチンが呼ばれる。検索ルーチンについては後述する。検索ルーチンの実行が終了してステップＳ８０５に戻ってくると次にステップＳ８０６に進む。尚、ステップＳ８０３においてポインタＰが指し示すリンクにｃｈｉｌｄが存在していなかったときは直接にステップＳ８０６に進む。

【０１８６】

ステップＳ８０６では、ポインタＰで指し示されたリンクがギアやカム等の受動部品を駆動する駆動リンクであるか否かが判定される。

【０１８７】

ポインタＰが受動部品を駆動する駆動リンクであったときは、ステップＳ８０７に進み、ポインタＰが指し示すリンクによって駆動される受動部品であるリンクを指し示すようにポインタＰが変更され、ステップＳ８０８において図３０の検索ルーチンが呼ばれる。検索ルーチンの実行が終了してステップＳ８０８に戻

ってくると次にステップＳ８０９に進む。尚、ステップＳ８０６において、ポインタＰが指し示すリンクが受動部品を駆動するリンクではないと判定されると、直接にステップＳ８０９に進む。

【０１８８】

このステップＳ８０９にまで進んだ段階では、リストＡが完成しており、このステップＳ８０９では、全リンクからリストＡにリストアップされたリンクを除いたリンクがリストＢに格納される。次いで、リストＡにリストアップされた全てのリンクとリストＢにリストアップされた全てのリンクとの間で干渉チェックが行なわれ（ステップＳ８１０）、干渉していたリンクの組のうち、リストＢにリストアップされた方のリンクがリストＢから削除される。

【０１８９】

図３０に示す検索ルーチンでは、先ずステップＳ９０１において、ポインタＰが指し示すリンクがリストＡに既にリストアップされているか否かが判定され、既にリストアップされていたときはそのままこの検索ルーチンを抜けてこの検索ルーチンが呼ばれた元のルーチンに戻る。ステップＳ９０１において、ポインタＰが指し示すリンクがリストＡにリストアップされていなかったときはステップＳ９０２に進み、ポインタＰが指し示すリンクがリストＡに追加される。ステップＳ９０３ではポインタＰが指し示すリンクにｃｈｉｌｄが存在するか否かが判定され、ｃｈｉｌｄが存在していたときにはステップＳ９０４に進んでポインタＰがそのｃｈｉｌｄのリンクを指し示すように変更され、ステップＳ９０５において再度この図３０に示す検索ルーチンが呼ばれる。このように、この検索ルーチンは何重にも重なって呼ばれることがあり、この検索ルーチンの実行が終了すると、その呼ばれた先(例えばステップＳ９０５で呼ばれたときはそのステップＳ９０５)に戻る。

【０１９０】

ステップＳ９０６では、ポインタＰが指し示すリンクにｂｒｏｔｈｅｒが存在するか否かが判定され、ｂｒｏｔｈｅｒが存在していたときは、ステップＳ９０７に進んでポインタＰがそのｂｒｏｔｈｅｒのリンクを指し示すように変更され、ステップＳ９０８において再度この検索ルーチンが呼ばれる。

【０１９１】

ステップＳ９０９では、ポインタＰで指し示されるリンクが、ギアやカム等の受動部品を駆動する駆動リンクであるか否かが判定され、駆動リンクであったときはステップＳ９１０に進んでポインタＰがその駆動リンクを指し示すように変更され、ステップＳ９１１においてこの検索ルーチンが再度呼ばれる。

【０１９２】

図３１は、図２９，図３０に示すルーチンの動作説明用のリンク機構モデルを示す図、図３２は、図３１に示すリンク機構モデルのデータ構造を示す図である。

【０１９３】

図３１に示すように、リンクＡはモータ、リングＢおよびリンクＣは、それぞれ、リンクＡ(モータ)の回転に伴って一緒に回転するシャフトおよびギア、リンクＤはリンクＣと噛合しているギアである。リンクＥはリンクＤ（ギア）の回転に伴って一緒に回転するシャフト、リンクＦはリンクＤ（ギア）と噛合しているギアである。

【０１９４】

データ構造上は、図３２に示すように、リンクＡとリンクＤとリンクＦが同格に並び、リンクＢはリンクＡのｃｈｉｌｄ、リンクＣはリンクＢのｂｒｏｔｈｅｒ、リンクＥはリンクＤのｃｈｉｌｄの関係にある。

【０１９５】

図３１,図３２に示すリンク機構モデルについて、図２９,図３０に示すルーチンの実行によりリストＡを作成することを考える。ここではリンクＡ（モータ）から検索が始められる。すなわち、図２９のステップＳ８０１においてリンクＡ

が指し示されるようにポインタＰが指定される。以下のかっこ内は、リストＡにリストアップされるリンクをあらわしている。

【０１９６】

（１）Ｐ＝Ａでスタート（Ａ）

（２）ＰがＡからそのｃｈｉｌｄＢに変更され、検索ルーチンが実行される（Ａ，Ｂ）

（３）ＰがＢからｂｒｏｔｈｅｒＣに変更され、検索ルーチンが実行される（Ａ，Ｂ，Ｃ）

（４）ＰがＣの受動部品であるＤに変更され、検索ルーチンが実行される（Ａ，Ｂ，Ｃ，Ｄ）

（５）ＰがＤからそのｃｈｉｌｄＥに変更され、検索ルーチンが実行される（Ａ，Ｂ，Ｃ，Ｄ，Ｅ）

（６）ｒｅｔｕｒｎでＰがＤに戻る。

【０１９７】

（７）ＰがＤからＤの受動部品であるＦに変更され、検索ルーチンが実行される（Ａ，Ｂ，Ｃ，Ｄ，Ｅ，Ｆ）

（８）ｒｅｔｕｒｎでＰがＤに戻る。

（９）ｒｅｔｕｒｎでＰがＣに戻る。

（１０）ｒｅｔｕｒｎでＰがＢに戻る。

（１１）ｒｅｔｕｒｎでＰがＡに戻る。

（１２）終了

以上の手順を踏んで、図３１に示す全てのリンク（リンクＡ～リンクＦ）がリストＡにリストアップされる。

【０１９８】

図３３は、１つのモータリンクの動きによって姿勢の変化を受けるグループ内での干渉可能性の評価方法の説明図である。

【０１９９】

モータリンクの動きによって干渉する可能性があるのは、図２６を参照して説明したような、そのモータリンクの動きによって姿勢に影響を受けるリンクのグループと姿勢に影響を受けないリンクのグループとの間のみでなく、図３１に示すように姿勢に影響を受けるグループの内部でも干渉が生じる可能性がある。そこで、そのグループの内部で干渉する可能性のあるリンクの組を発見するために、ここでは、シミュレーションに先立って、モータリンクを少しずつ動かしながら、そのモータリンクを動かすことによって姿勢に影響を受けるグループ内の全てのリンクどうしの間の干渉チェックを行う。このような総あたり的な干渉チェックによって干渉したリンクの組は実際のシミュレーションにあたっても相互間の距離を求める対象に加え、シミュレーション間隔Δｔを決定するための最短距離の候補の１つとする。こうすることによって正確なシミュレーションを行なうことができる。ただし、モータリンクを動かしたときに姿勢に影響を受けるグループ内において、例えば図３１に示すモータと歯車のように、常に接触しているリンクの組も存在する。そこでモータリンクを少しずつ動かして干渉チェックを行った際そのモータリンクの全ての姿勢で常に干渉・接触していたリンクの組、例えば歯車等で関係づけられたリンクの組はリンク間の距離を求める対象から排除される。

【０２００】

図３４は、モータリンクの動きによって姿勢に影響を受けるリンクグループ内で、シミュレーション時に距離を求めるリンクの組を抽出する手法の説明図である。

【０２０１】

図３４（Ａ）はある１つのモータの動きによって姿勢に変化を受けるグループを構成する全部品を示しており、ここでは一例として、図３４（Ｂ）に示すように、そのモータを－１８０°から１８０°まで36°おきに回転させ、各回転ご

とに干渉チェックを行なうこととする。この場合、モータの回転、リンク内の干渉チェックは１１回行なわれることになる。すると、図３４（Ｃ）に示すような、干渉したリンクの組と干渉回数のデータが得られる。干渉チェックは１１回行なわれており、リンクＢとリンクＥ、およびリンクＤとリンクＥは１１回干渉が発生していたため、図３４（Ｄ）に示すように、これら２つの組はシミュレーション時における距離測定の対象からは除外される。

【０２０２】

次に本発明の機構制御プログラム開発支援システムを構成する三次元機構モデルシミュレータの別の実施形態について説明する。

【０２０３】

図３５は、本実施形態におけるシミュレーション全体の処理の流れを示す図である。

【０２０４】

ここでは、制御プログラムとの同期処理（ステップＳ１００１）、モータ処理（ステップＳ１００２）、関節不良動作処理（ステップＳ１００３）、センサ処理（ステップＳ１００４）、および制御プログラムの実行時間ΔＴの決定処理（ステップＳ１００５）が繰り返し実行される。

【０２０５】

制御プログラムとの同期処理（ステップＳ１００１）に関しては、前述の実施形態と同一であるため（例えば図９，図２１を参照）、ここでは説明は割愛する。それ以外の処理に関しては、以下順次説明する。

【０２０６】

図３６は、モータ処理ルーチンを表わすフローチャートである。このフローチャートは、基本的には、前述した実施形態における、図１０の一点鎖線で囲った部分からなるモータ処理にモータの故障の概念を取り入れたものである。

【０２０７】

ステップＳ１１０１では、最短距離Ｄを求めるために先ずそのＤに∞が初期入力され、ステップＳ１１０２では、未処理のモータが存在しているか否かが判定され、未処理のモータ１つずつについてステップＳ１１０３～Ｓ１１１０の処理が行なわれる。

【０２０８】

即ち、先ず未処理のモータそれぞれについて、そのモータが故障しているか否かが判定される（ステップＳ１００３）。モータの故障の有無は、ユーザによりあらかじめ設定され、フラグとして保存されている。故障のモータについては処理は行なわれず、故障でないときは、制御プログラム側から送られてきた、今処理しようとしているモータに関するアクチュエータ信号に応じて、そのモータの姿勢変位量Δθが決定され（ステップＳ１００４）、Δθ＝０でなければ（ステップＳ１１０５）、そのモータのそれまでの姿勢θに今回の変位量Δθが加算され（ステップＳ１１０６）、そのモータリンクの移動に伴って移動するリンクの姿勢変位量を計算するリンク姿勢移動サブルーチン（図３７参照、後述する）が実行され（ステップＳ１１０７）、部品間最短距離ｄが求められ（ステップＳ１１０８）、ｄ＜Ｄか否かが判定され、ｄ＜Ｄのとき、すなわち、今回求めた最短距離ｄがこれまでに求めた最短距離Ｄよりも短かいときは（ステップＳ１１０９）、Ｄにｄを格納して最短距離Ｄを更新する（ステップＳ１１１０）。

【０２０９】

全てのモータに関し、今回の演算時間ΔＴの間の姿勢変更が終了すると、このモータ処理を終了する。

【０２１０】

図３７は、図３６に示すモータ処理ルーチンのステップＳ１１０７で実行されるリンク姿勢移動サブルーチンのフローチャートである。

【０２１１】

先ず、リンク姿勢移動（図３７のステップＳ１１０７でコールされたときは、

モータリンクの姿勢移動）を受けて（ステップＳ１２０１）、その姿勢移動のあったリンクに対する受動リンクに未計算の受動リンクが存在するか否か判定し（ステップＳ１２０２）、未計算の受動リンクが存在するときは、今回姿勢移動のあったリンクとそのリンクに対する受動リンクとの間にクラッチが存在するか否かを判定し（ステップＳ１２０３）、クラッチが存在するときは、そのクラッチ部品の姿勢がオンの位置にあるか否かを判定する（ステップＳ１２０４）。クラッチ部品の存在、およびクラッチ部品の姿勢とクラッチのオン、オフとの関係は、ユーザによりあらかじめ定義される。

【０２１２】

クラッチ部品が存在しないとき、あるいはクラッチ部品が存在していてもオン状態にあるときは、ステップＳ１２０５に進み、ギア、カム関係より受動部品の移動量を計算し、さらにステップＳ１２０６において、今回移動量を計算した受動部品を能動部品としたときのその能動部品に対する受動部品の移動量を求めるために、再度このリンク姿勢移動サブルーチンがコールされる。

【０２１３】

このようにして、このリンク姿勢移動サブルーチンは、図３０に示す検索ルーチンと同様、再帰的に実行され、この再帰的な実行により、１つのモータを動かしたとき他の部品の移動量が高速に求められる。

【０２１４】

図３８は、図３５のステップＳ１００３で実行される関節不良動作処理ルーチンのフローチャートである。

【０２１５】

ここでは、先ず、三次元機構モデルが画面表示され（ステップＳ１３０１）、その後、ユーザが新たに不良リンクを指定したか否か（ステップＳ１３０２）、未処理のユーザ指定不良リンクが存在するか否か（ステップＳ１３０５）、未処理の、干渉が起きたリンクが存在するか否か（ステップＳ１３１０）、および不良リンクが存在するか否か（ステップＳ１３１４）が判定される。

【０２１６】

ステップＳ１３０２において、ユーザが新たに不良リンクを指定したと判定されると、ステップＳ１３０３に進み、その新たに指定された不良リンクを駆動しているモータを検索し、そのモータとそのモータの移動方向をリストに追加する（ステップＳ１３０４）。

【０２１７】

ステップＳ１３０５において未処理のユーザ指定不良リンクが存在すると判定されると、ステップＳ１３０６に進み、その不良リンクを駆動しているモータの移動方向が逆転したか否かが判定され、逆転していないときはステップＳ１３０７に進んでそのモータの姿勢を旧姿勢に戻し、そのモータの姿勢が旧姿勢に戻されたことに対応してリンクの姿勢を旧姿勢に戻すために、図３７に示すリンク姿勢移動サブルーチンがコールされる（ステップＳ１３０８）。一方、ステップＳ１３０６においてそのモータの移動方向が逆転したと判定されると、ステップＳ１３０９に進み、そのモータを不良リンクのリストから削除する。

【０２１８】

こうすることにより、不良リンクがある一方向に移動しようとしたときに何らかの原因で引っ掛かって動かなくなり、その不良リンクが一旦逆方向に移動することによりその引っ掛かりが解除されて正常に戻るという動作がシミュレートされる。

【０２１９】

ステップＳ１３１０において、未処理の、干渉が起きたリンクが存在する旨判定されると、ステップＳ１３１１に進み、その不良リンクを駆動しているモータの検索が行われ、そのモータの姿勢が旧姿勢に戻され（ステップＳ１３１２）、そのモータによって駆動される他のリンクの姿勢を旧姿勢に戻すべく、図３７に示すリンク機構サブルーチンがコールされる（ステップＳ１３１３）。

【０２２０】
これにより、他のリンクと干渉してそれ以上動かすことのできない状態をシミュレートすることができる。

【０２２１】
ステップＳ１３１４において不良リンクが存在すると判定されると、その三次元機構モデルを画像上に再描画する（ステップＳ１３１５）。こうすることにより、ステップＳ１３０１における描画と合わせ、シミュレーション画面上で見ると不良リンクを受動部品に持ったモータおよびそのモータにより駆動される全リンクの動きがカクカクし、いかにもひっかかった様子が表現される。

【０２２２】
尚、ここに示した例では、モータの姿勢を元に戻すことによって関節不良を実現したが、関節値の上限あるいは下限を一時的に現在値に設定することで関節不良を実現することも可能である。

【０２２３】
図３９は、図３５のステップＳ１００４で実行されるセンサ処理ルーチンのフローチャートである。前述の実施形態では、センサはオン、オフの信号を出力するオン／オフセンサに限られていたが、ここでは、センサとして、オン／オフセンサのほか、ポテンショメータおよびエンコーダも存在し、さらにセンサの不良も考慮されている。

【０２２４】
ステップＳ１４０１では、未処理のセンサの有無が判定され、未処理のセンサが存在すると、ステップＳ１４０２に進み、その未処理のセンサがオン／オフセンサであるか否かが判定される。オン／オフセンサであったときはステップＳ１４０３において、オン／オフセンサ処理が行なわれ、そのオン／オフセンサを処理した結果のセンサ値（オンあるいはオフ）が出力される。

【０２２５】
ステップＳ１４０２において、その未処理のセンサがオン／オフセンサではないと判定されると、ステップＳ１４０４に進み、そのセンサがポテンショメータであるかエンコーダであるかが判定され、ポテンショメータであると判定されるとステップＳ１４０５に進んでポテンショメータ前処理が行なわれた後、エンコーダであると判定されると直接に、ステップＳ１４０６のエンコーダ処理に進み、その結果としてのセンサ値が出力される。

【０２２６】
図４０は、図３９のステップＳ１４０３で実行されるオン／オフセンサ処理ルーチンのフローチャートである。

【０２２７】
このオン／オフセンサ処理ルーチンではそのオン／オフセンサの故障もシミュレートされており、先ずステップＳ１５０１，１５０３でそのオン／オフセンサがオンの状態のまま故障しているかあるいはオフの状態のまま故障しているかが判定され、オン状態のまま故障しているときは出力予定値としてオンが設定され（ステップＳ１５０２）、オフ状態のまま故障しているときは出力予定値としてオフが設定され（ステップＳ１５０４）、ステップＳ１５０８に進んでその出力予定値がセンサ値として出力される。そのオン／オフセンサがオン状態あるいはオフ状態で故障しているか否かはユーザによりあらかじめ設定される。

【０２２８】
そのオン／オフセンサが故障していないときはステップＳ１５０５に進み、そのセンサリンクの姿勢から出力予定値（オンあるいはオフ）が決定され（ステップＳ１５０５）、次いで性能劣化による出力予定値変更処理（ステップＳ１５０６）、およびチャッタリングによる出力予定値変更処理（ステップＳ１５０７）が実行される。

【０２２９】
図４１は、図４０のオン／オフセンサ処理ルーチンのステップＳ１５０６で実

行されるセンサ性能劣化処理ルーチンのフローチャートである。このルーチンでは、オン／オフセンサが故障（出力がオンまたはオフに固定）までいかなくても、応答性能が劣化し、そのオン／オフセンサの出力をオンまたはオフに変化させる状態が生じた後、暫らくは前の状態を保ち、遅れてオンまたはオフに変化する状態がシミュレートされる。

【０２３０】

ステップＳ１６０１では、旧出力予定値ｄと出力予定値とが等しいか否かが判定され、変化していたときはステップＳ１６０２に進み出力予定値がオンかオフかが判定される。出力予定値がオンのときはステップＳ１６０３に進み、センサ出力がオフからオンに変化するときの遅れ時間がＴｄに設定され、一方出力予定値がオフのときはステップＳ１６０４に進み、センサ出力がオフからオンに変化するときの遅れ時間がＴｄに設定される。

【０２３１】

ステップＳ１６０５ではＴｄ＞０か否か判定され、Ｔｄ＞０のときは、ステップＳ１６０６に進んでＴｄから制御プログラムの実行時間ΔＴが減算され、ステップＳ１６０７において出力予定値が反転される（出力予定値がオンのときはオフ、出力予定値がオフのときはオン）。

【０２３２】

ステップＳ１６０８では、旧出力予定値ｄに今回の出力予定値が格納される。

【０２３３】

図３５に示すように、センサ処理Ｓ１００４には繰り返し実行され、したがって図４１に示すセンサ性能劣化処理も繰り返し実行され、Ｔｄ＞０のときは、この処理が繰り返されるたびにＴｄからΔＴが減算されＴｄ≦０に達するとセンサが変化することになる。

【０２３４】

図４２は、図４０のオン／オフセンサ処理ルーチンのステップＳ１５０７で実行される、チャッタリングによる出力予定値変更処理ルーチンのフローチャートである。このルーチンではオン／オフセンサがオンからオフ、あるいはオフからオンに変化するときのチャッタリングがシミュレートされる。

【０２３５】

ステップＳ１７０１では、旧出力予定値ｃと出力予定値とが等しいか否かが判定され、変化していたときはステップＳ１７０２に進み、出力予定値がオンかオフかが判定される。出力予定値がオンのときはステップＳ１７０３に進み、センサ出力がオフからオンに変化するときのチャッタリング時間がＴｃに設定され、一方出力設定値がオフのときはステップＳ１７０４に進み、センサ出力がオンからオフに変化するときのチャッタリング時間がＴｃに設定される。

【０２３６】

ステップＳ１７０５では、旧出力予定値ｃに出力予定値が設定される。

【０２３７】

ステップＳ１７０６では、Ｔｃ＞０か否かが判定され、Ｔｃ＞０のときはステップＳ１７０７に進んでＴｃから制御プログラムの実行時間ΔＴが減算され、ステップＳ１７０８において０～９９の値をとる乱数が発生されてその発生した乱数があらかじめ設定された発生確率（０～９９のうちのいずれかの値）と比較され、乱数＞発生確率のときは出力予定値のオン、オフが反転される（ステップＳ１７０９）。

【０２３８】

このようにして、Ｔｃ＞０の間、確率的にオン、オフが変化するチャッタリングがシミュレートされる。

【０２３９】

次に、図３９のステップＳ１４０６で実行されるエンコーダ処理について説明する。

【０２４０】

図４３は、エンコーダの変位（角度等）に対する出力波形の一例を示す図、図４４は、エンコーダの、各種の出力波形を示す図である。

【０２４１】

ここでは、図４３に示すように、エンコーダの出力波形は、波長Ｌの周期的な関数ｆ（θ）をとるものと考える。ただし関数ｆ（θ）は、図４４に示すように各種の形をとることができる。ここで波長Ｌは、関数ｆ（θ）の一周期をあらわし、例えば、このエンコーダがロータリ式のエンコーダであって回転角度１．０度ごとに一周期の信号を出力するエンコーダであるときは、Ｌ＝１．０（度）となる。

【０２４２】

ここで、変位θを電気角φ＝２πθ／Ｌに変換し、関数ｆ（θ）を電気角φであらわしたときｇ（φ）と表記するものとして、以下、エンコーダ機能の原理を説明する。

【０２４３】

前回行なったシミュレーション時のエンコーダの電気角をφ、今回すなわち前回行なったシミュレーションから制御プログラムの演算時間ΔＴだけ進んだ時刻の電気角をφ、エンコーダの電気角速度をωとすると、前回から今回までの間のシミュレーションサイクルにおけるエンコーダの移動量はωΔＴとなり、

$$\phi = \phi_o + \omega \Delta T$$

が成立する。

【０２４４】

しかし、エンコーダはパルス数を測定するため、今回のシミュレーションサイクルにおけるωΔＴが２πより大きい場合、単純にｇ（φ）を出力するわけにはいかず、所定のパルス数のパルスを出力した後にｇ（φ）を出力する必要がある。

【０２４５】

図４５は、エンコーダの出力処理ルーチンのプログラムを示す図である。

【０２４６】

ここでは先ず前回のシミュレーション終了時のエンコーダ出力ｇ（ϕ_o）を先ず出力し（ステップＳ１８０１）、今回のシミュレーション終了時の電気角φとの差分φ－ϕ_oが$\phi - \phi_o > \pi/2$であるか（ステップＳ１８０２）、あるいは$\phi_o - \phi > \pi/2$であるか（ステップＳ１８０４）が判定され、$\phi_o - \phi > \pi/2$のときはステップＳ１８０３に進んで、ϕ_oにπ／２が加算され、ステップＳ１８０６において５０μｓｅｃ待機した後、ステップＳ１８０１に戻って新たなϕ_oに対するｇ（ϕ_o）が出力される。$\phi_o - \phi > \pi/2$のときは、ステップＳ１８０３に進みϕ_oからπ／２が減算され、その後は同様にステップＳ１８０６において５０μｓｅｃ待機した後、ステップＳ１８０１に戻って新たなϕ_oに対するｇ（ϕ_o）が出力される。

【０２４７】

すなわち、ここでは$|\phi - \phi_o| > \pi/2$のときは、$|\phi - \phi_o| \leqq \pi/2$になるまで、５ｋＨｚの周波数を上限として（ステップＳ１８０６）、φがπ／２ずつ変化したときの離散的な周期信号が出力される。

【０２４８】

$|\phi - \phi_o| \leqq \pi/2$に達すると、ステップＳ１８０７に進み、今回のシミュレーションにおけるエンコーダの最終値であるｇ（φ）が出力され、次回のシミュレーションのために今回のシミュレーションにおけるφがϕ_oに格納される（ステップＳ１８０８）。

【０２４９】

このエンコーダ信号は、制御プログラムとは独立して動作するカウンタ回路によって受信されることを想定しており、したがってエンコーダ信号発信時に制御プログラムが停止していても問題はない。

【０２５０】

また、図４５のフローチャートは、パルス信号をｇ（φ）の１つのみ出力するタイプのエンコーダを表現した例であるが、ｇ（φ）の他にｇ（φ＋π／２）を出力することにより、Ａ相、Ｂ相の二相信号を出力するタイプのエンコーダも表現することができる。

【０２５１】

さらにｇ（φ）のほかにφ－φ$_0$の正負の符号をオン／オフで出力することにより、アップ／ダウン信号を出力するタイプのエンコーダも表現することができる。

【０２５２】

ここで、上記のシミュレーションに先立って、パルスの波形（図４４参照）や分解能や電気角φがゼロになる位置をあらわすオフセットを入力することでエンコーダの特性がユーザにより定義される。

【０２５３】

図４６は、図３９に示す処理ルーチンのステップＳ１４０６で実行されるエンコーダ処理ルーチンのフローチャートである。

【０２５４】

ここではエンコーダの故障についても考慮されており、エンコーダに故障があったときの三次元機構モデルのシミュレーションのために、あらかじめユーザにより以下の故障フラグが設定される。

【０２５５】

故障フラグ０：正常
１：Ａ相がV_{max}で故障
２：Ａ相がV_{min}で故障
３：Ｂ相がV_{max}で故障
４：Ｂ相がV_{min}で故障
５：アップ／ダウン信号故障（Ｈｉｇｈに固定）
６：アップ／ダウン信号故障（Ｌｏｗに固定）

図４６のステップＳ１９０１では、先ず変位θが電気角φに変換され、次いで、故障フラグが１であるか否か（ステップＳ１９０２）、２であるか否か（ステップＳ１９０３）が判定され、それらの何れでもないときは、ステップＳ１９０４に進んで、図４５に示すｇ（φ）出力処理ルーチンが実行される。一方、ステップＳ１９０２において故障フラグが１であると判定されるとステップＳ１９０５に進み、V_{max}で故障していることに対応してｇ（π／４）が出力され、一方、ステップＳ１９０３において故障フラグが２であると判定されるとステップＳ１９０６に進みV_{min}で故障していることに対応してｇ（３π／４）が出力される。

【０２５６】

ステップＳ１９０７では、このエンコーダが２相出力タイプのエンコーダであるか否か、ステップＳ１９０８では、このエンコーダがアップ／ダウン信号を出力するタイプのエンコーダであるか否かが判定され、いずれでもない、単純にｇ（φ）のみを出力するタイプのエンコーダのときはそのままこのルーチンを抜ける。

【０２５７】

ステップＳ１９０７においてこのエンコーダが２相出力タイプのエンコーダであると判定されると、ステップＳ１９０９において故障フラグが３であるか否か、ステップＳ１９１０において故障フラグが４であるか否かが判定される。それらのいずれでもなかったときは、ステップＳ１９１１に進み、図４５におけるφをφ＋π／２に置き換えた出力処理ルーチンが実行される。ステップＳ１９０９において故障フラグが３であると判定されるとステップＳ１９１２に進み、Ｂ相がV_{max}で故障していることに対応してｇ（π／４）が出力され、これと同様に、ステップＳ１９１０において故障フラグが４であると判定されるとステップＳ１９１３に進み、Ｂ相がV_{min}で故障していることに対応してｇ（３π／４）が

出力される。

【０２５８】

また、ステップＳ１９０８においてこのエンコーダがアップ／ダウン信号出力タイプのエンコーダであると判定されると、ステップＳ１９１４に進み故障フラグが５であるか否かが判定され、ステップＳ１９１５では故障フラグが６であるか否かが判定される。それらのいずれでもないときはステップＳ１９１６に進みそのエンコーダの移動方向が判定され、アップ方向のときはステップＳ１９１７に進んで‘Ｈｉｇｈ’が出力され、ダウン方向のときはステップＳ１９１８に進んで‘Ｌｏｗ’が出力される。一方、ステップＳ１９１４において故障フラグが５であると判定されるとステップＳ１９１７に進み、エンコーダの移動方向のいかんによらず‘Ｈｉｇｈ’が出力され、これと同様に、ステップＳ１９１５において故障フラグが６であると判定されるとステップＳ１９１８に進み、エンコーダの移動方向のいかんによらず‘Ｌｏｗ’が出力される。

【０２５９】

このようにしてエンコーダを表現するとともにエンコーダの故障を表現することができる。ただし、エンコーダの場合、オン／オフセンサの場合と異なり、処理に入った直後に故障中を示すのではなく、センサ信号を出力する際に故障フラグの判定、処理が行なわれる。

【０２６０】

図４７は、ポテンショメータの関数値の一例を示す図である。

【０２６１】

ここでは、ポテンショメータを、図４４（ｄ）に示す三角波を出力する、分解能が極端に低いエンコーダとして実現している。例えば図４３に示す波長Ｌとして３６０度を設定すれば、$\theta_{max}-\theta_{min}$＝１８０度の変位で出力電圧Ｖが最小値$V_{min}$から最大値$V_{max}$まで変換するポテンショメータを表現することができる。

【０２６２】

図４８は、図３９に示すセンサ処理ルーチンのステップＳ１４０５で実行されるポテンショメータ前処理ルーチンのフローチャートである。

【０２６３】

ポテンショメータの場合、図４７に実線で示すように、θ_{min}以下でのθでは出力電圧Ｖは最低値に固定され、θ_{max}以上のθでは出力電圧Ｖは最大値に固定されることがあり、この図４８ではそのようなポテンショメータを実現している。

【０２６４】

すなわち、ステップＳ２００１において$\theta > \theta_{max}$を満足するか否かが判定され、それを満足するときはステップＳ２００２に進んでθがθ_{max}に固定される。

【０２６５】

またこれと同様に、ステップＳ２００３において$\theta < \theta_{min}$を満足するか否かが判定され$\theta < \theta_{min}$を満足するときはステップＳ２００４に進んでθがθ_{min}に固定される。

【０２６６】

ポテンショメータの場合、このような前処理が行なわれた後、図４６に示すエンコーダ処理が実行される（図３９参照）。

【０２６７】

このタイプのポテンショメータの場合、ユーザは、θ_{min}，θ_{max}および，V_{min}，V_{max}を入力すると、システムは自動的に三角波ｆ（θ）の波長、振幅、オフセットに変換する。

【０２６８】

尚、ストッパがないポテンショメータなど、出力が周期的に変化するポテンショメータの場合は、分解能が低い三角波や鋸波を出力するエンコーダとして取り扱われる。

【０２６９】

図４９は、図３５のステップＳ１００５において実行されるΔＴ決定処理ルーチンのフローチャートである。

【０２７０】

ここでは、先ずステップＳ２１０１において、性能劣化（応答遅れ、あるいはチャッタリング）を起こしたセンサが存在するか否かが判定され、そのようなセンサが存在しないときは、図３６に示す処理で求めた部品間の最短距離Ｄに応じたΔＴが決定され（ステップＳ２１０２）、一方、性能劣化を起こしたセンサが存在するときは、ΔＴとしてとり得る値の最小値が設定される。

【０２７１】

ここで、性能劣化を起こしたセンサが存在するときにΔＴを最小値に設定するのは、性能劣化の表現精度を高め、正確なシミュレーションを行なうためである。なお正常なセンサの場合、遅れ時間はゼロに設定されているため、問題なく動作する。

【０２７２】

図５０は、干渉チェックによる関節可動範囲の検索法の説明図である。

【０２７３】

先ずユーザによって、どの方向の制限値を設定したいのかということと、どの部品とどの部品が接触することにより制限が起きるのかということが選択され、その後、関節可動範囲の検索が行なわれる。

【０２７４】

この関節可動範囲の検索においては、関節の姿勢ｘを初期値x_oからΔｘずつ増加させ、その度に干渉チェックを行なう。すると図５０のように、ｘが$x_o+n\Delta x$では干渉が発生せず、次の$x_o+(n+1)\Delta x$で干渉が起こる位置が判明する。続いて、$\Delta x' = \Delta x / 10$などのようにΔｘより小さい移動量Δｘ’を用いて同様にｘを$x_o+\Delta x$からΔｘ’ずつ増加させることにより、ｘが$x_o+n\Delta x+n'\Delta x'$では干渉が発生せず、$x_o+n\Delta x+(n'+1)\Delta x'$で干渉が発生する位置が判明する。

【０２７５】

このような処理を数回繰り返すことにより、高速に、かつ求めようとする精度でｘの制限値$x_o+n\Delta x+n'\Delta x'+n''\Delta x''$……を求めることができる。ただしユーザが指定した二つの部品が、関節をどのように動かしていても干渉しない位置関係であることも考えられるので、一回目の干渉位置検索の際には検索数ｎに上限を設け無限ループを防ぐことが好ましい。

【０２７６】

図５１は、干渉チェックによるカム関係の検索方法の説明図、図５２は、干渉チェックによるカム関係検索ルーチンのフローチャートである。

【０２７７】

ここでは、あらかじめ、ユーザにより、駆動部品、受動部品、駆動部品の初期姿勢、受動部品の初期姿勢、駆動部品の最終姿勢が指定される。

【０２７８】

図５２に示すルーチンは、先ず、ϕ，ϕに、それぞれ、駆動部品初期姿勢、受動部品初期姿勢が設定され（ステップＳ２２０１）、次いで、受動部品の非接触姿勢が検索されてϕに格納される（ステップＳ２２０２）。

【０２７９】

ステップＳ２２０３では、今回求めた、駆動部品の姿勢ϕと、受動部品の非接触姿勢ϕとのペアが登録される。駆動部品の姿勢ϕが$\Delta\phi$だけ移動され（ステップＳ２２０４）、駆動部品の姿勢ϕが最終姿勢になるまで、$\Delta\phi$ずつ移動された駆動部品に対する受動部品の非接触姿勢ϕの演算（ステップＳ２２０２）、登録（ステップＳ２２０３）が繰り返される。

【０２８０】

図５３は、図５２に示すカム関係検索ルーチンのステップＳ２２０２で実行さ

れる受動部品の非接触姿勢検索ルーチンのフローチャートである。

【０２８１】

先ずステップＳ２３０１において、受動部品の現在の姿勢φがφ$_o$に設定され、カウンタとしてのｎに初期値ゼロが設定される。

【０２８２】

ステップＳ２３０２では駆動部品と受動部品とが干渉しているか否かが判定され、干渉していないときは、現在の姿勢φが非接触姿勢となる（ステップＳ２３０３）。

【０２８３】

ステップＳ２３０２で干渉している旨判定されると、ステップＳ２３０３においてｎがインクリメントされ、受動部品の姿勢φとしてφ$_o$＋ｎΔφが設定され（ステップＳ２３０４）、再度干渉しているか否かが判定される（ステップＳ２３０５）。まだ干渉していたときは、今度は受動部品の姿勢φとしてφ$_o$－ｎΔφが設定され（ステップＳ２３０６）、干渉しているか否かがもう一度判定される（ステップＳ２３０７）。まだ干渉していたときは、ステップＳ２３０８に進み、ｎがｎの上限か否か判定され、ｎの上限であったときは検索失敗とされる（ステップＳ２３０９）。ｎが未だ上限に達していなかったときは、ステップＳ２３０３に戻りｎが１だけインクリメントされ、以下同様にして干渉チェックが行われる。

【０２８４】

ステップＳ２３０５で干渉していないと判定されると、ステップＳ２３１０に進み、φがΔφだけデクリメントされ、Δφが適当な整数Ｎで割り算されてΔφとして小さな値が設定される（ステップＳ２３０３）。次いで、φが、上記のようにして小さな値に設定されたΔφだけインクリメントされながら（ステップＳ２３１３）、干渉しているか否かが判定され（ステップＳ２３１２）、干渉した状態から干渉していない状態に変化するとステップＳ２３１４に進んで所定の精度まで検索したか否かが判定され、未だ精度が不充分なときはステップＳ２３１０に戻ってφからΔφが減算され、ステップＳ２３１１においてΔφとしさらに小さい値が設定され、φをさらに小さい値であるΔφずつインクリメントしながら干渉チェックが行なわれる（ステップＳ２３１２，Ｓ２３１３）。

【０２８５】

このようにして所定の精度まで検索が行なわれると（ステップＳ２３１４）、これにより非接触姿勢が定まる（ステップＳ２３０３）。

【０２８６】

ステップＳ２３０７において干渉していないと判定されたときもステップＳ２３１０～Ｓ２３１４の処理と同様の処理が実行され、受動部品の非接触姿勢が求められる。尚、ステップＳ２３０７において干渉していないと判定されたときは、ステップＳ２３１０に相当するステップにおいてφはΔφだけインクリメントされ、ステップＳ２３１３に相当するステップにおいてφはΔφだけデクリメントされる。

【０２８７】

図５４、図５２に示すカム関係検索ルーチンのステップＳ２２０２で実行される受動部品の非接触姿勢検索ルーチンのフローチャートである。このルーチンは、受動部品に重力あるいはバネ付勢力が作用している場合に実行されるルーチンである。受動部品に作用する重力あるいはバネの付勢方向はユーザによりあらかじめ設定される。図５３は、負方向に力が加わっている場合のフローチャートを示している。

【０２８８】

図５４のルーチンでは先ず、ステップＳ２４０１において、受動部品の現在の姿勢φがφ$_o$に設定され、カウンタとしてのｎに初期値ゼロが設定される。

【０２８９】

ステップＳ２４０２では、駆動部品と受動部品が現在干渉しているか否かが判

定される。既に干渉しているときは、駆動部品の今回の姿勢変化によって受動部品が干渉したものであるため、受動部品の姿勢を重力あるいはバネ付勢力にさからって変化させ（ステップＳ２４０４～Ｓ２４０７）、干渉していない姿勢をみつけた後、φからΔφを減算することで（ステップＳ２４０８）、一旦干渉した状態に戻す。

【０２９０】

一方、ステップＳ２４０２で干渉していない旨判定されるときは、駆動部品の今回の姿勢変化によって受動部品が干渉していない状態に変化したものであるため、重力あるいはバネ付勢力の作用をシミュレートして受動部品が能動部品に干渉した状態を見つける（ステップＳ２４０９～Ｓ２４１３）。

【０２９１】

ステップＳ２４１４に進んだとき、Δφのきざみでぎりぎり干渉した状態にあり、ΔφをＮで割り算することにより、Δφとして小さな値を設定し、φをΔφずつインクリメントしながら非干渉姿勢が検索される（ステップＳ２４１５，Ｓ２４１６）。ステップＳ２４１７では所定の精度まで検索が行なわれたか否かが判定され、未だ精度が不充分なときはステップＳ２４０８に進んでφからΔφが減算されて干渉した状態に戻され、ステップＳ２４１４においてΔφがＮで割り算されてΔφとしてさらに小さい値が設定されて検索が繰り返される。

【０２９２】

このようにして所定の精度まで検索が行なわれると（ステップＳ２４１４）、これにより非接触位置が決定される（ステップＳ２４１８）。

【０２９３】

このようにして、重力あるいはバネ付勢力が作用しない場合であっても作用する場合であっても、カム関係を決定することができる。

【０２９４】

図５５は、溝にピンが挿入されている溝関係を設定する手法の説明図である。

【０２９５】

図５５に示すような溝関係を設定しようとしたとき、ピンは常に接触した状態にあるため、非干渉位置を求めることはできず、このような溝関係を前述したカム関係の検索法を用いて求めることは困難である。

【０２９６】

そこで、ここでは、図５５（ｃ）に示すように、ピンの中心に半径ｒの微小な円筒を一時的に取り付け、この円筒と溝との間の距離ｄを測定することで溝関係を求める。すなわち、図５３における「干渉しているか？」の判定に代えて「ｄ＜Ｒ－ｒか？」を判定することで溝関係を検索することができる。

【０２９７】

【発明の効果】

以上説明したように、本発明によれば、機構を物理的に製作することなく、機構制御用のプログラムのデバックを行なうことができ、その機構制御プログラムの開発の短縮化、コスト低減化に大きく寄与することになる。

【図面の簡単な説明】

【図１】

内部に機構制御プログラム開発支援システムが構築されたコンピュータシステムの外観図である。

【図２】

図１に外観を示すコンピュータシステムのハードウェア構成図である。

【図３】

プログラム記憶媒体内に記憶されたプログラムの構成を示す図である。

【図４】

機構制御プログラム開発支援システムの一例が構築されたコンピュータネットワークの一例を示す図である。

【図５】

図４に外観を示すコンピュータネットワークを構成するコンピュータシステムのハードウェア構成図である。

【図６】

機構制御プログラム開発支援システムの概念図である。

【図７】

センサやモータの定義時の処理を示す３Ｄモデルシミュレータの模式図である。

【図８】

シミュレーション時の３Ｄモデルシミュレータの処理を示す模式図である。

【図９】

シミュレーション時における３Ｄモデルシミュレータと制御プログラムの動作タイミングチャートを示す図である。

【図１０】

シミュレーション時のシミュレータのフローを示すフローチャートである。

【図１１】

図１０に示すシミュレータの処理フローの一部分（図１０に一点鎖線で囲った部分）に代えて採用することのできる部分フローを示すフローチャートである。

【図１２】

図１２は、モータとして定義されるリンクの選び方を示す模式図である。

【図１３】

センサとして定義されるリンクの選び方を示す模式図である。

【図１４】

センサとして定義されるリンクの選び方を示す模式図である。

【図１５】

モータの種類を示す模式図である。

【図１６】

モータを一次遅れ系とみなしたときの、モータの回転速度ωの初期変化を示す図である。

【図１７】

シミュレータにおける、一次遅れを考慮したモータ駆動の処理を示すフローチャートである。

【図１８】

干渉チェックを用いたセンサの定義方法の説明図である。

【図１９】

干渉チェックを用いたセンサの定義方法の説明図である。

【図２０】

干渉チェックを用いたセンサの定義方法の説明図である。

【図２１】

シミュレータと制御プログラムとの間で同期をとるために送受信される信号のタイミングチャートである。

【図２２】

３Ｄモデルの動作との同期をとるための同期化プログラムを示すフローチャートである。

【図２３】

同期を実現するための、シミュレータ側の処理を示すフローチャートである。

【図２４】

干渉可能性の評価方法の説明図である。

【図２５】

部品間の距離ｄとシミュレーション間隔Δｔとの関係の各種の例を示す図である。

【図２６】

部品のグループ分けの説明図である。

【図２７】
１つの３Ｄモデル内に複数のモータリンクが定義されている場合のグループ分けの説明図である。
【図２８】
グループ分けした部品の再グループ化の説明図である。
【図２９】
モータリンクの動きによって姿勢に影響を受けるリンクのグループとそのモータリンクの動きによっては姿勢に影響を受けないグループとに分けるルーチンを示すフローチャートである。
【図３０】
検索ルーチンのフローチャートである。
【図３１】
図２９，図３０に示すルーチンの動作説明用のリンク機構モデルを示す図である。
【図３２】
図３１に示すリンク機構モデルのデータ構造を示す図である。
【図３３】
１つのモータリンクの動きによって姿勢の変化を受けるグループ内での干渉可能性の評価方法の説明図である。
【図３４】
モータリンクの動きによって姿勢に影響を受けるリンクグループ内で、シミュレーション時に距離を求めるリンクの組を抽出する手法の説明図である。
【図３５】
本実施形態におけるシミュレーション全体の処理の流れを示す図である。
【図３６】
モータ処理ルーチンを表わすフローチャートである。
【図３７】
リンク姿勢移動サブルーチンのフローチャートである。
【図３８】
関節不良動作処理ルーチンのフローチャートである。
【図３９】
センサ処理ルーチンのフローチャートである。
【図４０】
オン／オフセンサ処理ルーチンのフローチャートである。
【図４１】
センサ性能劣化処理ルーチンのフローチャートである。
【図４２】
チャッタリングによる出力予定値変更処理ルーチンのフローチャートである。
【図４３】
エンコーダの変位（角度等）に対する出力波形の一例を示す図である。
【図４４】
エンコーダの、各種の出力波形を示す図である。
【図４５】
エンコーダの出力処理ルーチンのプログラムを示す図である。
【図４６】
エンコーダ処理ルーチンのフローチャートである。
【図４７】
ポテンショメータの関数値の一例を示す図である。
【図４８】
ポテンショメータ前処理ルーチンのフローチャートである。
【図４９】
ΔＴ決定処理ルーチンのフローチャートである。

【図５０】
干渉チェックによる関節可動範囲の検索法の説明図である。
【図５１】
干渉チェックによるカム関係の検索方法の説明図である。
【図５２】
干渉チェックによるカム関係検索ルーチンのフローチャートである。
【図５３】
受動部品の非接触姿勢検索ルーチンのフローチャートである。
【図５４】
受動部品の非接触姿勢検索ルーチンのフローチャートである。
【図５５】
溝にピンが挿入されている溝関係を設定する手法の説明図である。
【符号の説明】
１００，４００，５００　コンピュータシステム
１０１，４０１，５０１　本体部
１０１ａ，４０１ａ，５０１ａ　フロッピィディスク装填口
１０１ｂ，４０１ｂ，５０１ｂ　ＣＤＲＯＭ装填口
１０２，４０２，５０２　ＣＲＴディスプレイ
１０２ａ，４０２ａ，５０２ａ　表示画面
１０３，４０３，５０３　キーボード
１０４，４０４，５０４　マウス
２１０，４１０　ＣＤＲＯＭ
２１１，４１１　ハードディスク
２１２，４１２　フロッピィディスク
２２０，４２０バス
２２１，４２１　中央演算処理装置（ＣＰＵ）
２２２，４２２　ＲＡＭ
２２３，４２３　ハードディスクコントローラ
２２４，４２４　フロッピィディスクドライバ
２２５，４２５　ＣＤＲＯＭドライバ
２２６，４２６　マウスコントローラ
２２７，４２７　キーボードコントローラ
２２８，４２８　ディスプレイコントローラ
３００　プログラム記憶媒体
３１０　機構制御プログラム開発支援プログラム
３１１　三次元機構モデルシミュレーションプログラム
３１２　同期化プログラム
４２９　モデム

【書類名】　　要約書

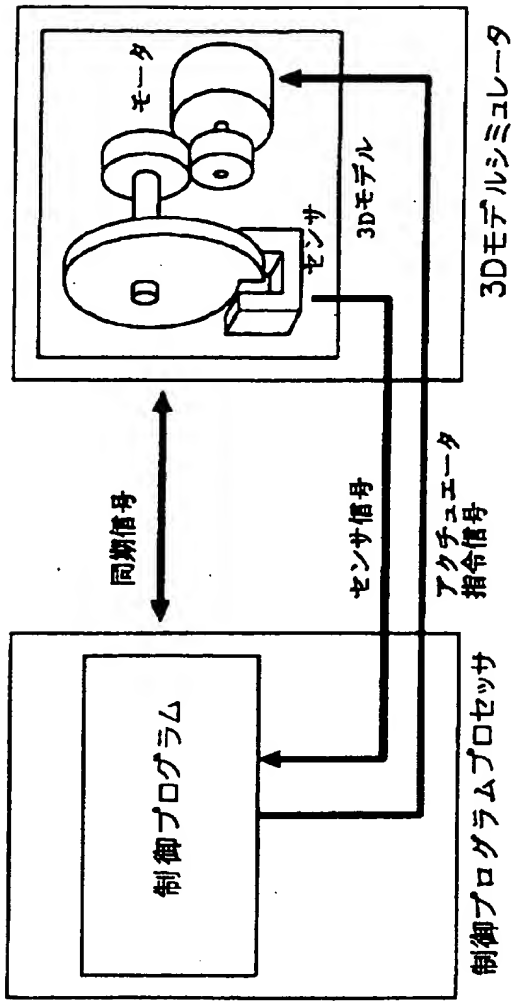
【要約】

【課題】機構を制御する制御プログラムの開発を支援する。

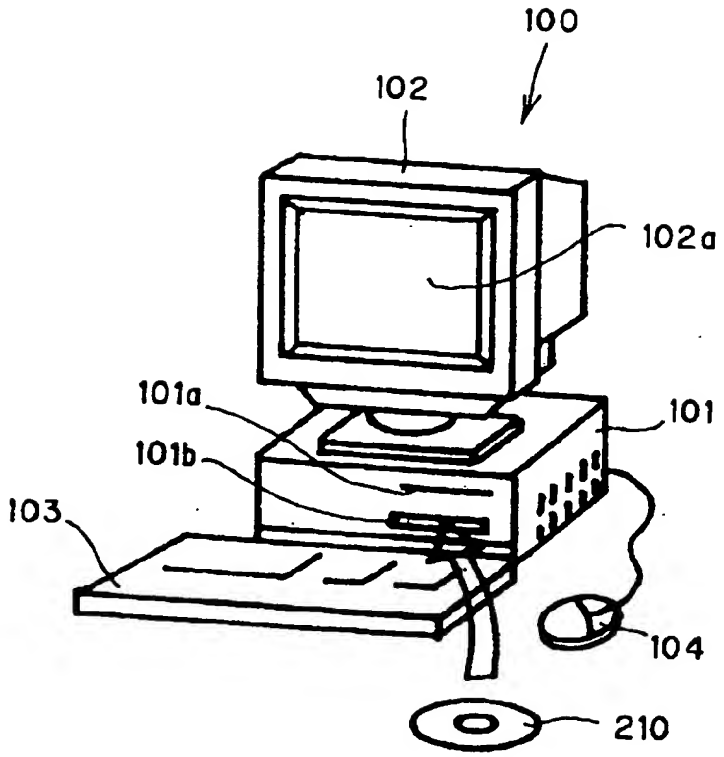
【解決手段】内部に構築された、アクチュエータおよびセンサを含む複数の部品からなる三次元機構モデルを動作させる三次元機構モデルシミュレータと、三次元機構モデルシミュレータ内に構築された三次元機構モデルの動作を制御する制御プログラムを、三次元機構モデルシミュレータ内で動作する三次元機構モデルとの同期をとりながら実行する制御プログラムプロセッサとを備える。

【選択図】　　図６

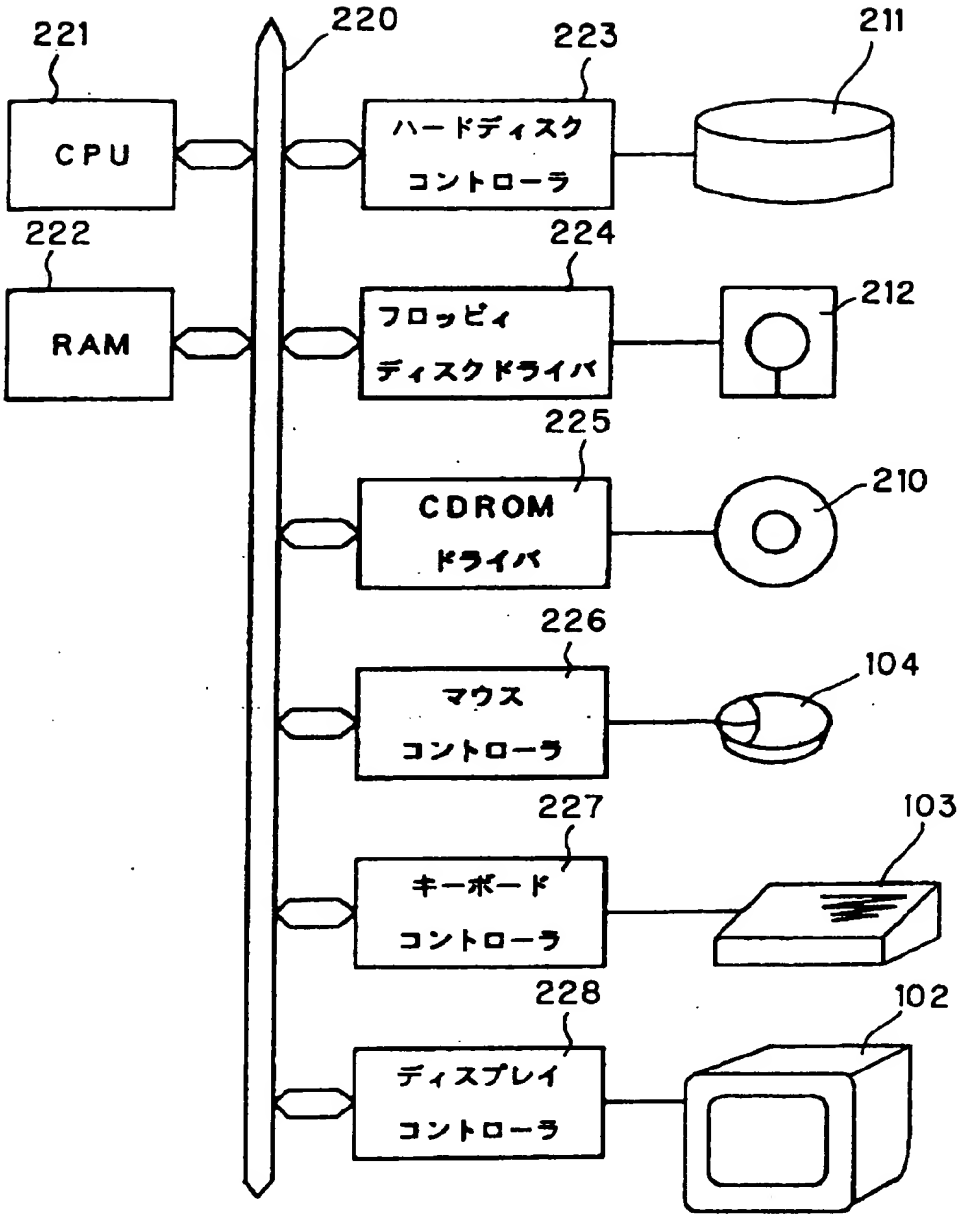
【書類名】　図　面
【図６】


モータ
センサ
3Dモデル
3Dモデルシミュレータ
同期信号
センサ信号
アクチュエータ
指令信号
制御プログラム
制御プログラムプロセッサ

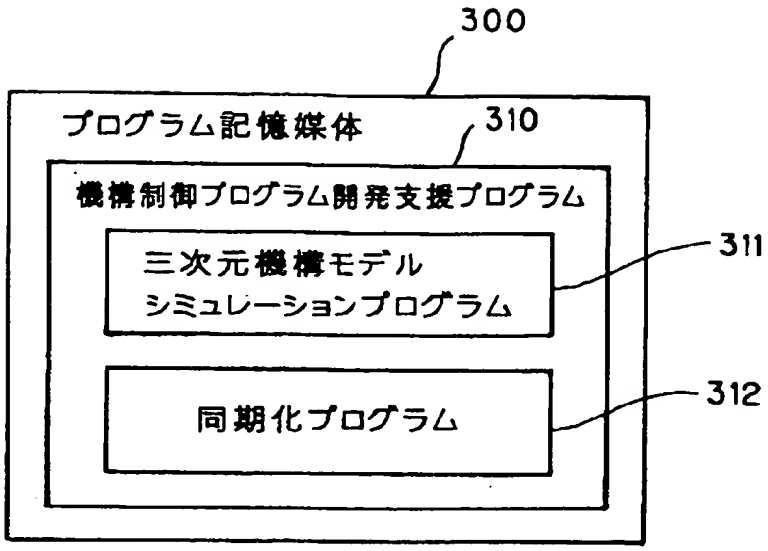
【図１】



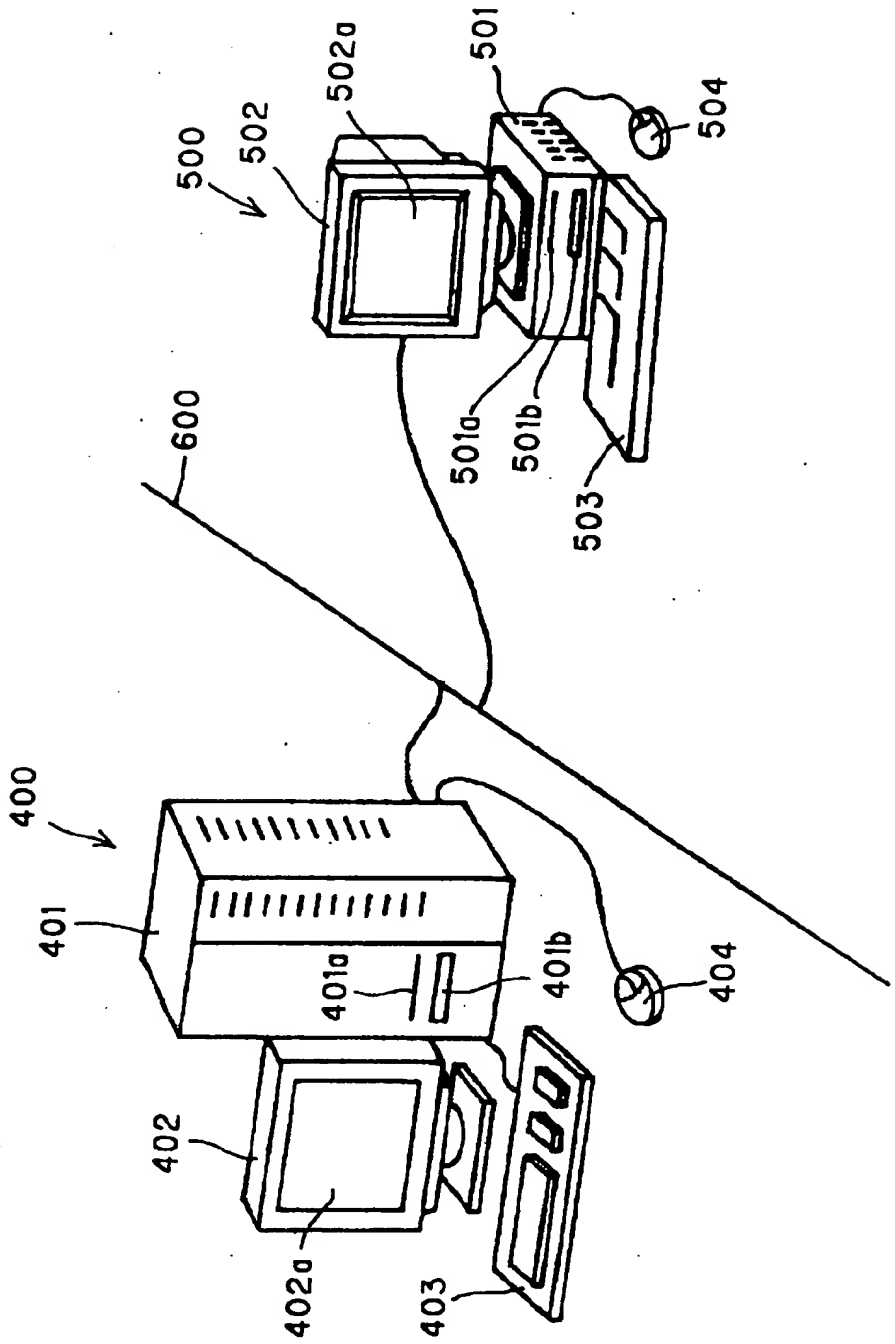
【図２】


221
220
223
211
CPU
ハードディスク
コントローラ
222
224
RAM
フロッピィ
ディスクドライバ
212
225
CDROM
ドライバ
210
226
マウス
コントローラ
104
227
103
キーボード
コントローラ
228
102
ディスプレイ
コントローラ

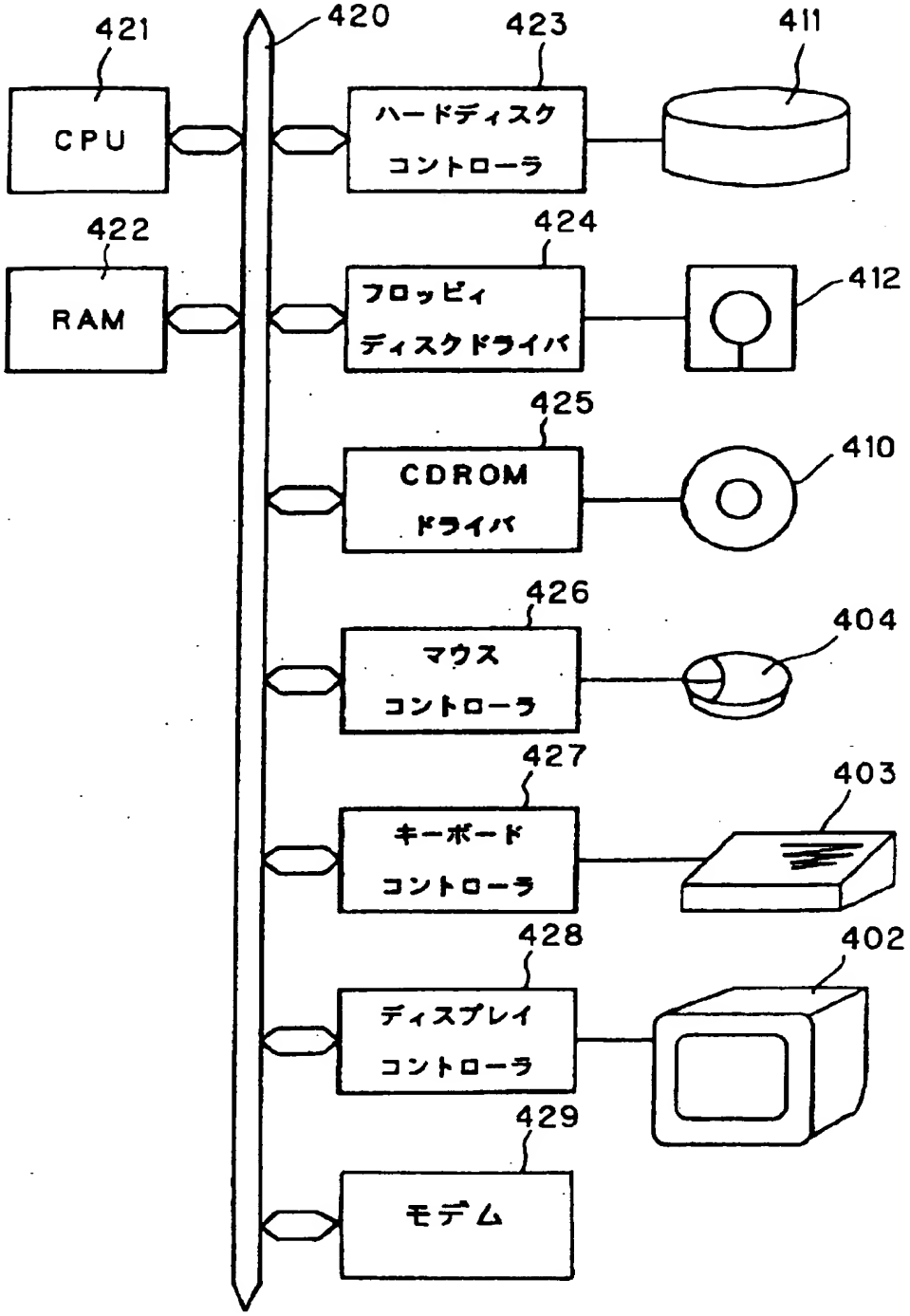
【図３】


300
プログラム記憶媒体
310
機構制御プログラム開発支援プログラム
三次元機構モデル
シミュレーションプログラム
311
同期化プログラム
312

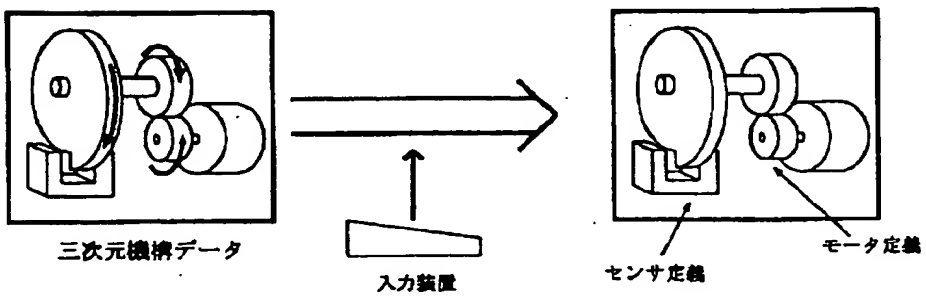
【図４】


500
502
502a
501
504
501a
501b
503
600
400
401
401a
401b
404
402
402a
403

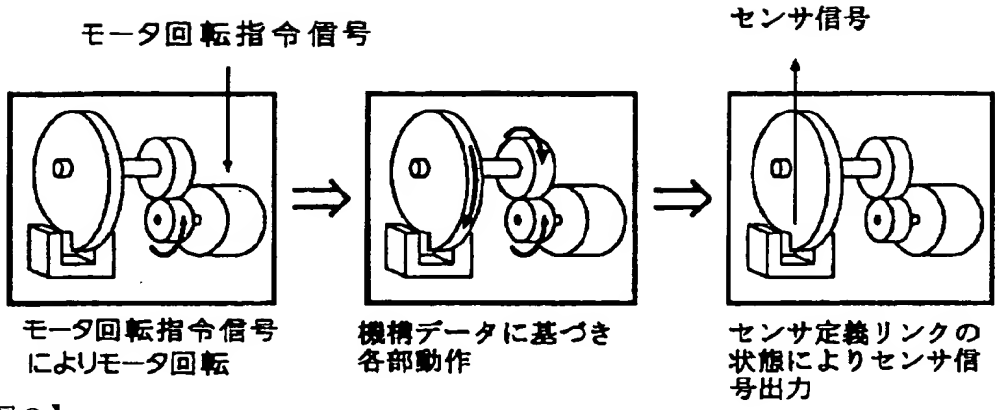
【図５】


421
420
423
411
CPU
ハードディスク
コントローラ
422
424
412
RAM
フロッピィ
ディスクドライバ
425
410
CDROM
ドライバ
426
404
マウス
コントローラ
427
403
キーボード
コントローラ
428
402
ディスプレイ
コントローラ
429
モデム

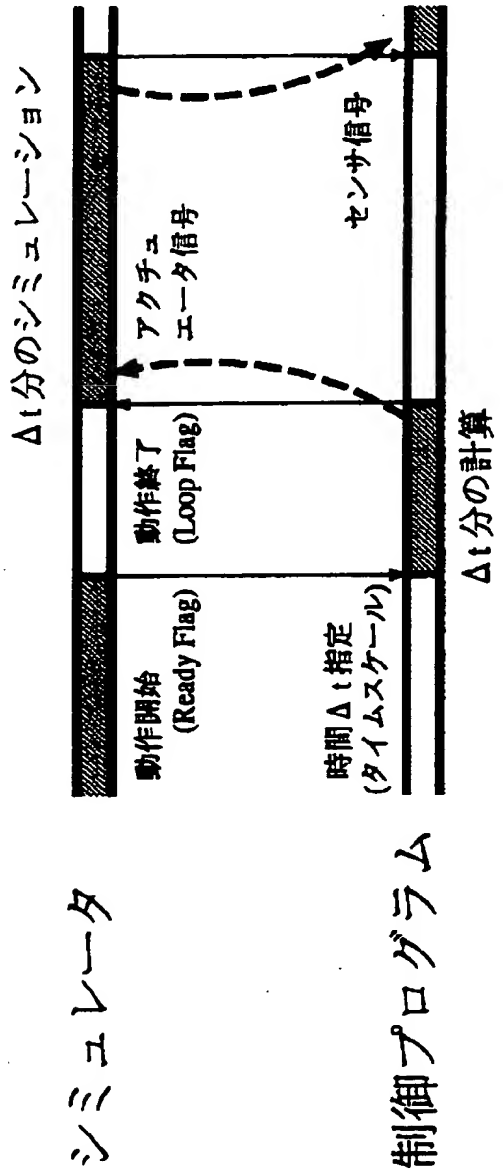
【図７】


三次元機構データ
入力装置
センサ定義
モータ定義

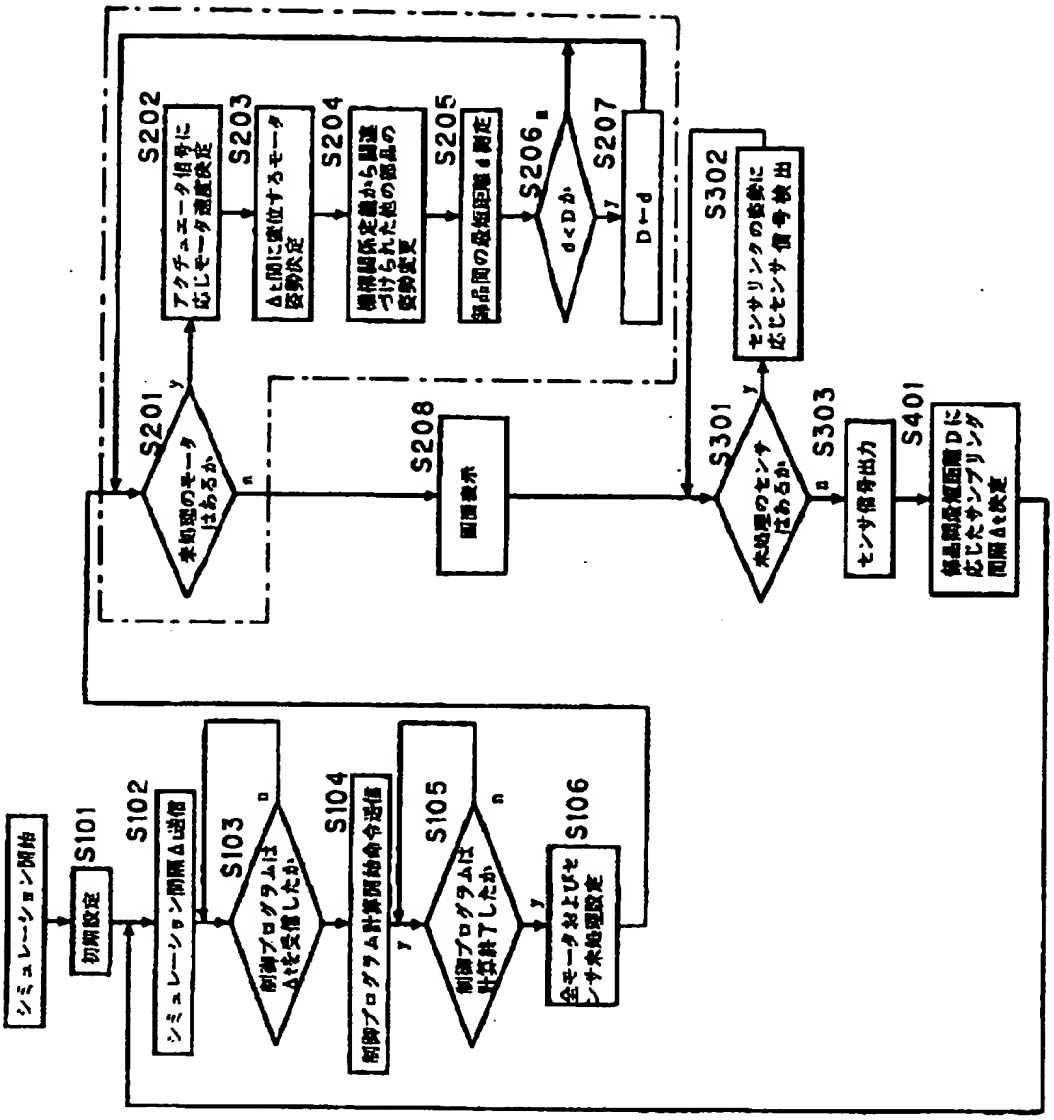
【図８】


モータ回転指令信号
センサ信号
モータ回転指令信号
によりモータ回転
機構データに基づき
各部動作
センサ定義リンクの
状態によりセンサ信
号出力

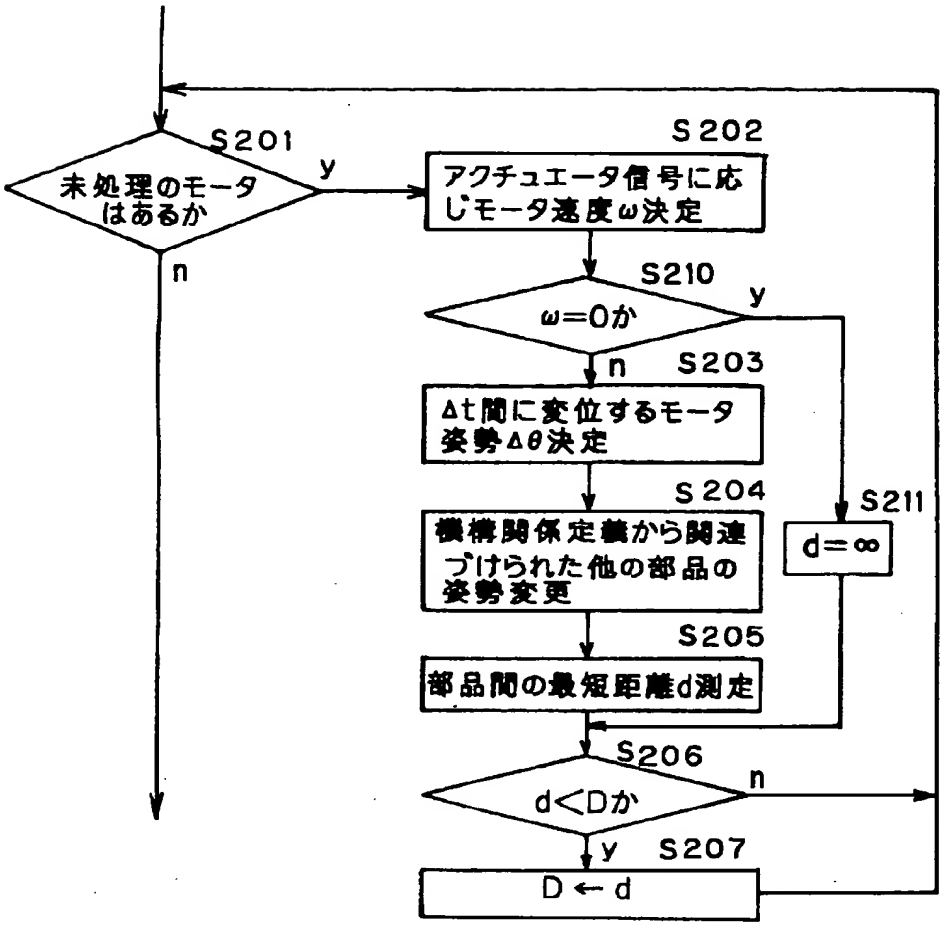
【図９】


Δt分のシミュレーション
センサ信号
アクチュエータ信号
動作終了
(Loop Flag)
動作開始
(Ready Flag)
時間Δt指定
(タイムスケール)
Δt分の計算
シミュレータ
制御プログラム

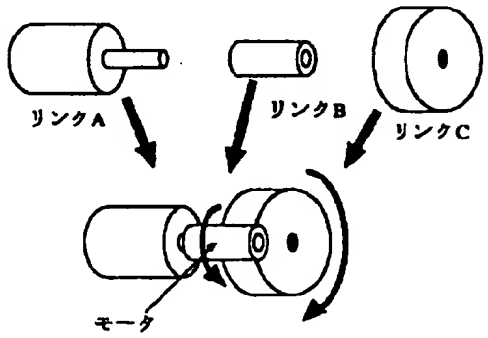
【図１０】


シミュレーション開始
S101
初期設定
S102
シミュレーション開始Δt進行
S103
制御プログラムはΔtを完了したか
S104
制御プログラム計算開始命令送信
S105
制御プログラムは計算終了したか
S106
全モータおよびセンサ出力設定
S201
未処理のモータはあるか
S202
アクチュエータ信号に応じモータ速度決定
S203
Δt間に変位するモータ変位決定
S204
機構関係定義から関連づけられた他の部品の変位変更
S205
隣接間の最短距離d測定
S206
d<Dか
S207
D←d
S208
距離要求
S301
未処理のセンサはあるか
S302
センサリンクの名前に応じセンサ信号検出
S303
センサ信号出力
S401
機構関係距離Dに応じたサンプリング間隔Δt決定

【図１１】

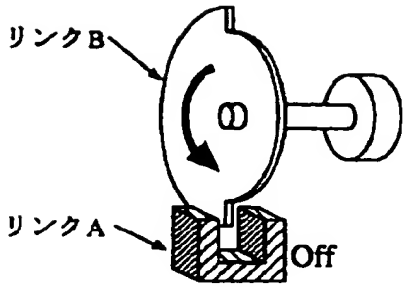

S201
未処理のモータはあるか
y
n
S202
アクチュエータ信号に応じモータ速度ω決定
S210
ω=0か
y
n
S203
Δt間に変位するモータ姿勢Δθ決定
S204
機構関係定義から関連づけられた他の部品の姿勢変更
S211
d=∞
S205
部品間の最短距離d測定
S206
d<Dか
n
y
S207
D←d

【図１２】

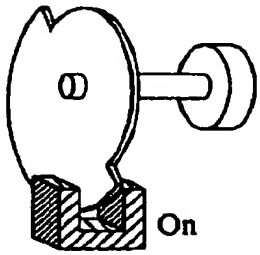

リンクA
リンクB
リンクC
モータ

【図１３】

(A)

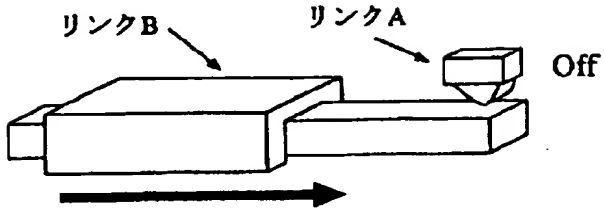


(B)

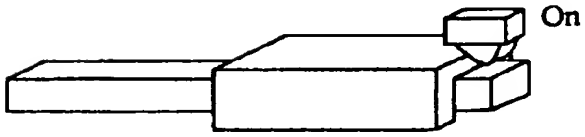


【図１４】

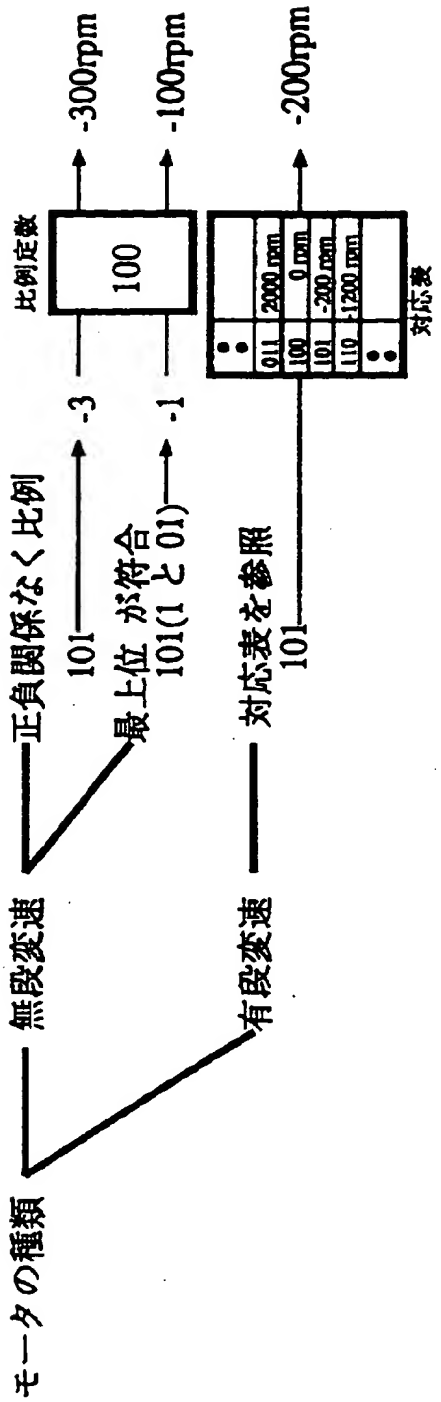
(A)



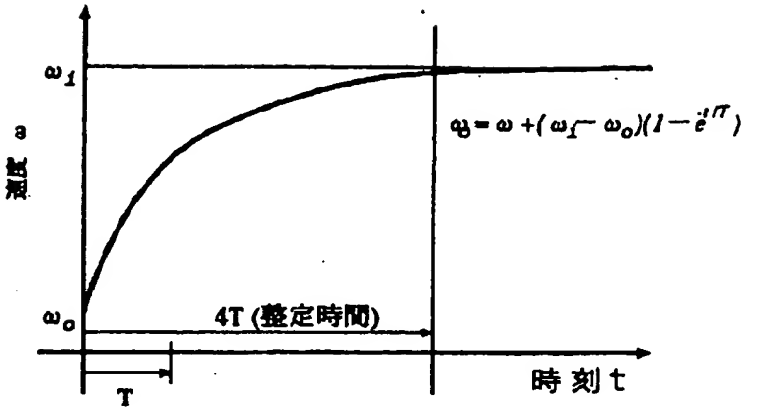
(B)



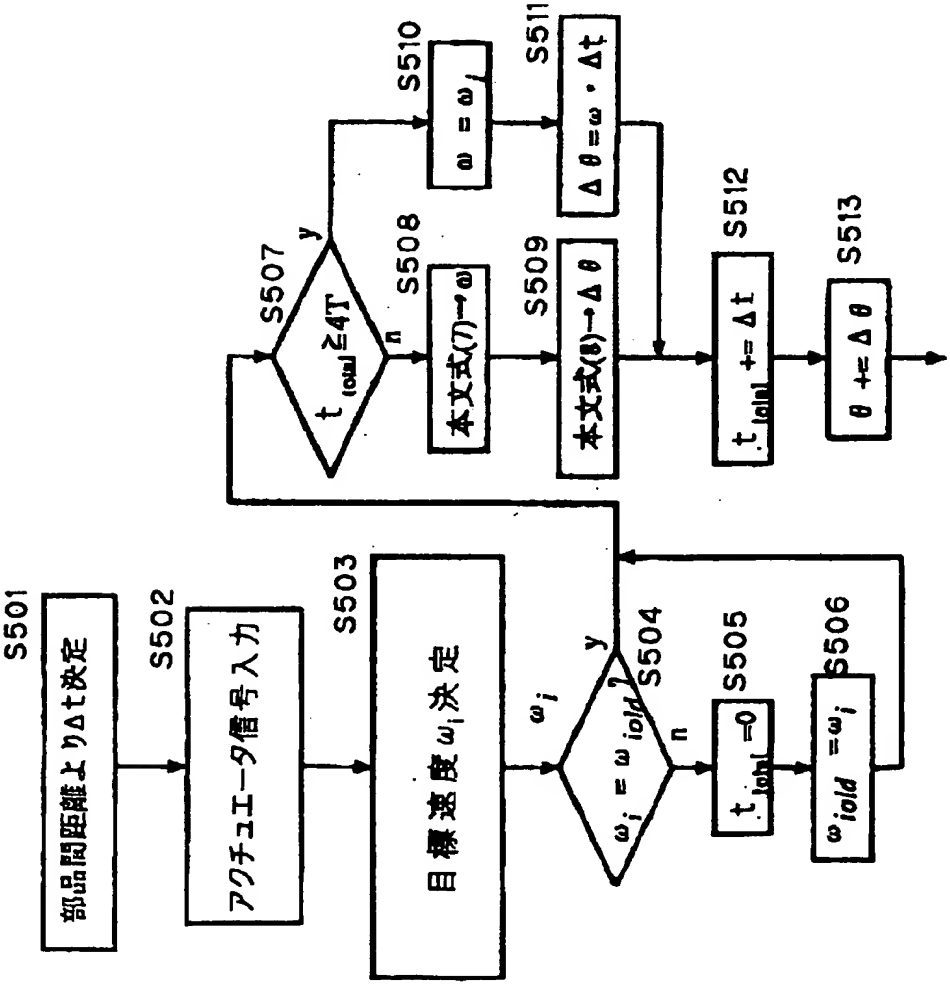
【図１５】


モータの種類
無段変速
有段変速
正負関係なく比例
101
-3
最上位 が符合
101(1 と 01)
-1
対応表を参照
101
比例定数
100
-300rpm
-100rpm
-200rpm
対応表
011
100
101
110
2000 rpm
0 rpm
-200 rpm
-1200 rpm

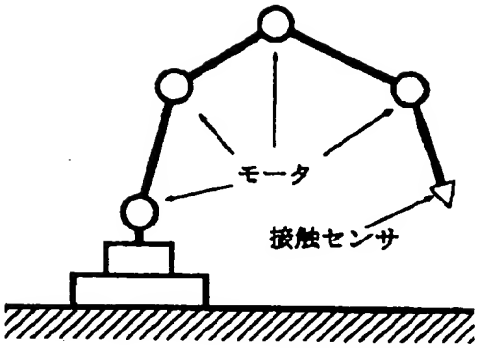
【図１６】


速度 ω
ω1
ω0
ω = ω +(ω1− ω0)(1−e^−t/T)
4T (整定時間)
T
時刻 t

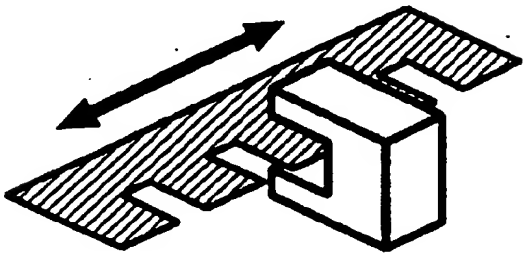
【図１７】


S501
部品間距離よりΔt決定
S502
アクチュエータ信号入力
S503
目標速度ωi決定
S504
S505
S506
S507
S508
S509
S510
S511
S512
S513
y
n
本文式(7)→ω
本文式(8)→Δθ
ω = ω1
Δθ=ω・Δt
θ += Δθ

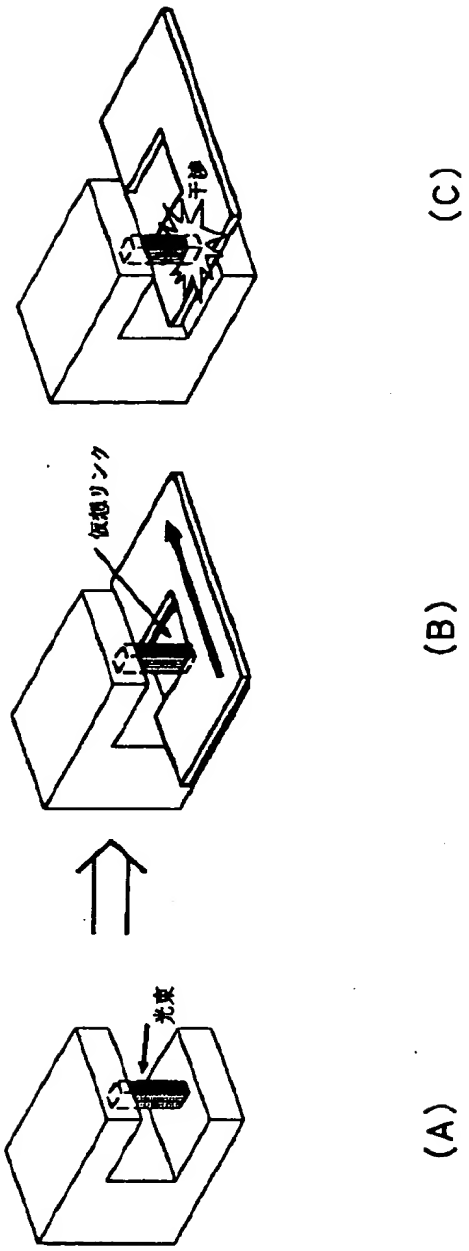
【図１８】



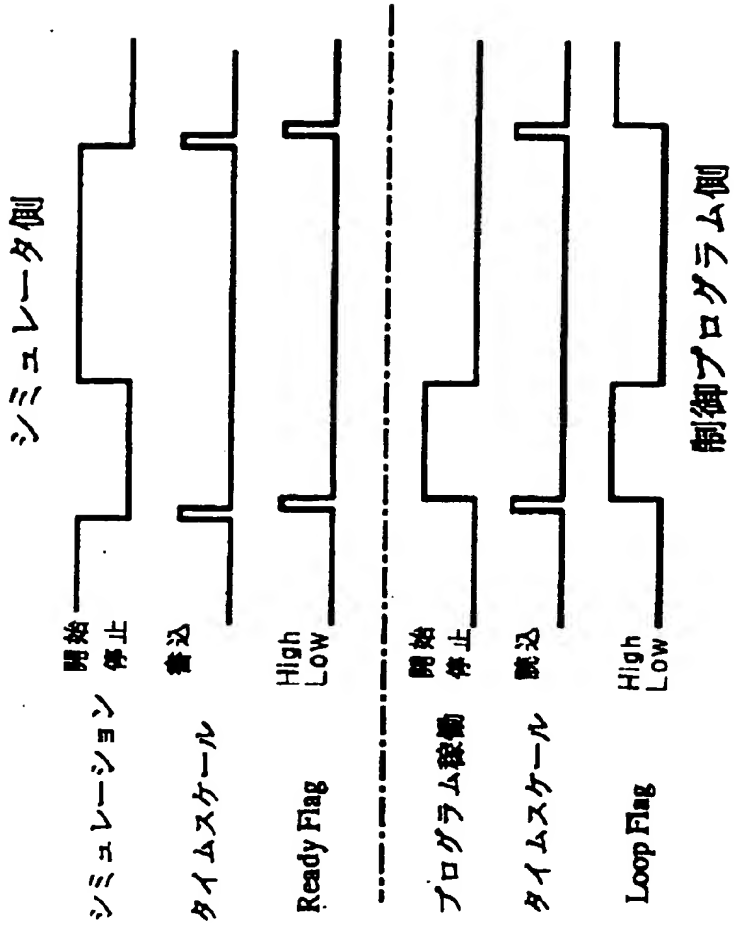
【図１９】



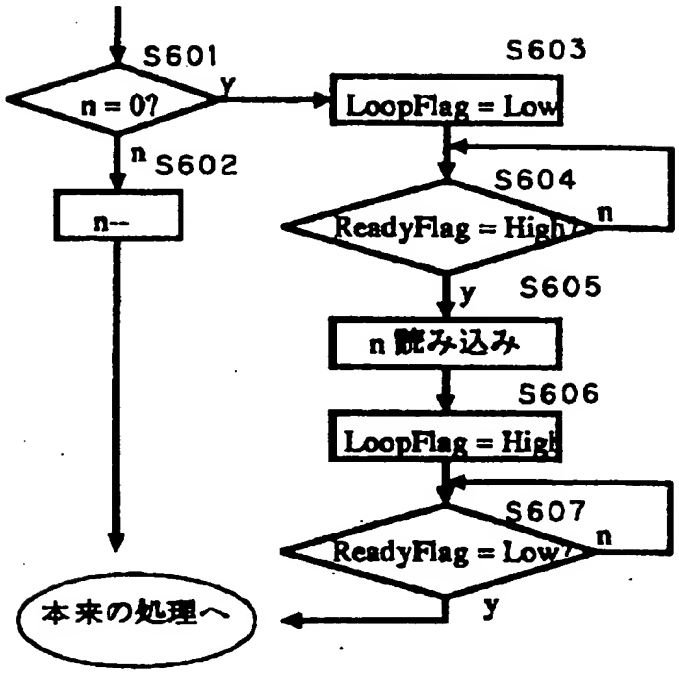
【図２０】



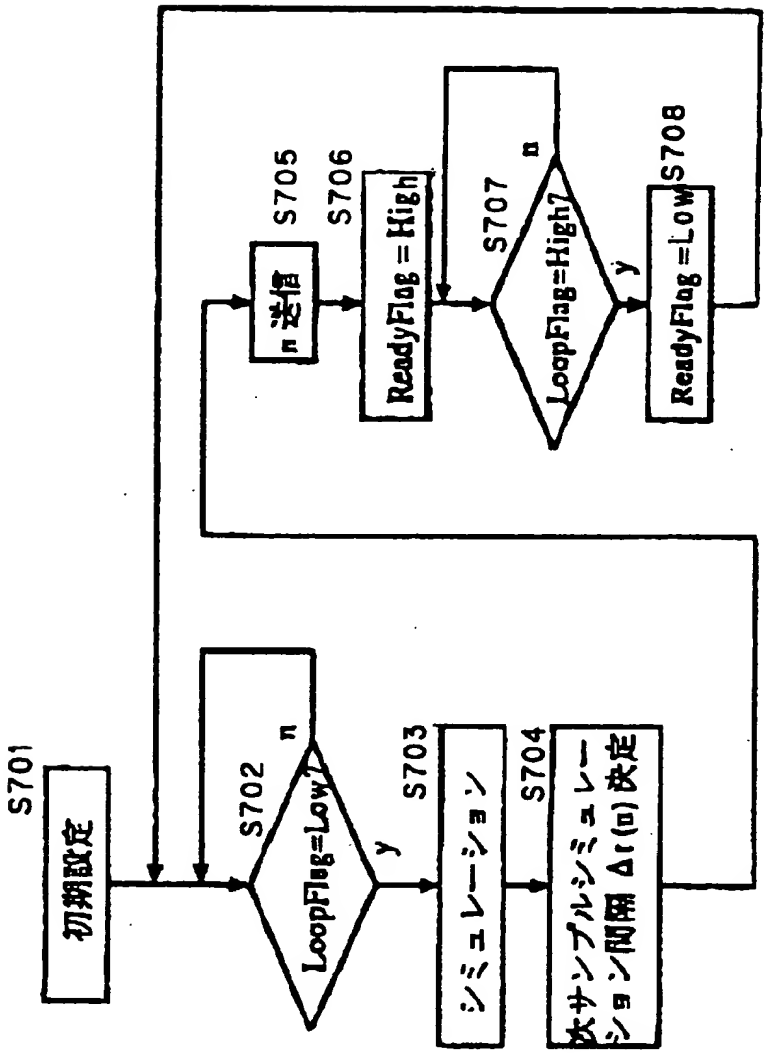
【図２１】


シミュレータ側
制御プログラム側
シミュレーション
開始
停止
タイムスケール
書込
Ready Flag
High
Low
プログラム稼働
開始
停止
タイムスケール
読込
Loop Flag
High
Low

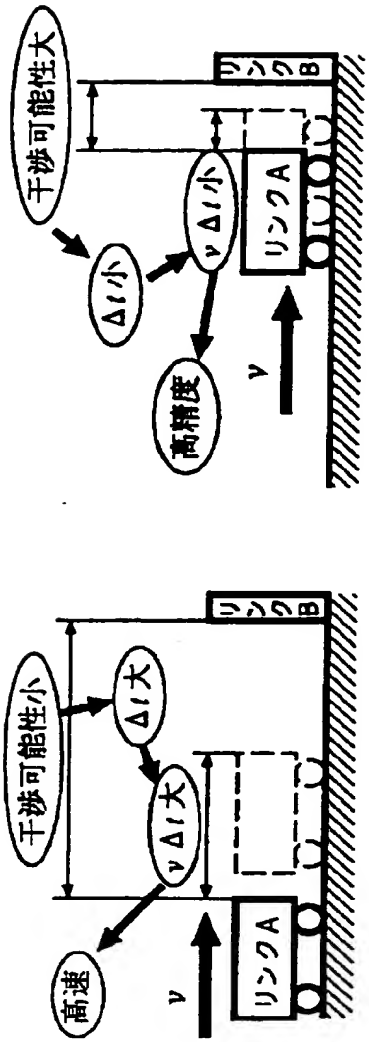
【図２２】


S601
n = 0?
y
n
S602
n--
S603
LoopFlag = Low
S604
ReadyFlag = High?
n
y
S605
n 読み込み
S606
LoopFlag = High
S607
ReadyFlag = Low?
n
y
本来の処理へ

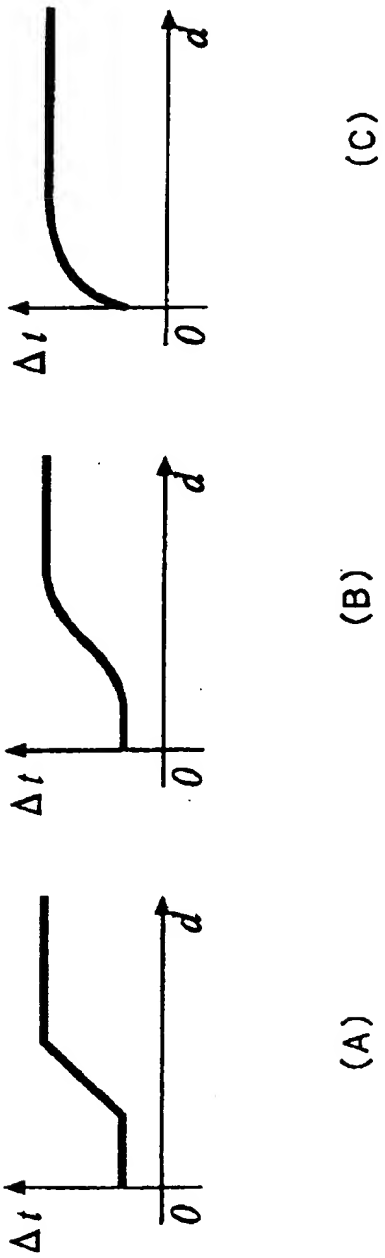
【図２３】


S701
初期設定
S702
LoopFlag=Low?
n
y
S703
シミュレーション
S704
次サンプルシミュレーション間隔 Δr(n)決定
S705
n 送信
S706
ReadyFlag = High
S707
LoopFlag=High?
n
y
S708
ReadyFlag =Low

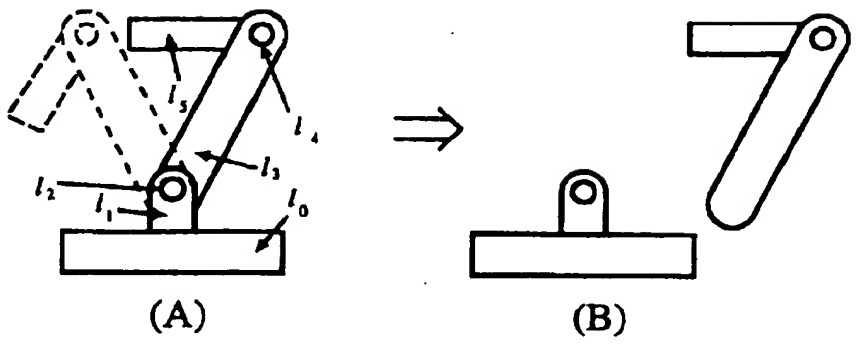
【図２４】


干渉可能性大
Δt小
ν Δt小
高精度
ν
リンクA
リンクB
(B)
干渉可能性小
Δt大
ν Δt大
高速
ν
リンクA
リンクB
(A)

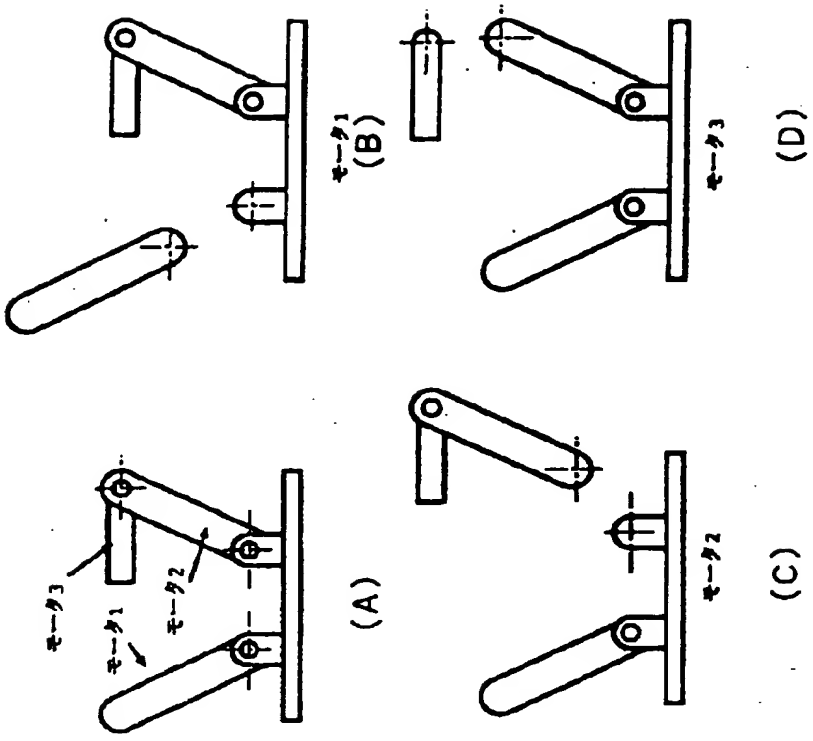
【図２５】



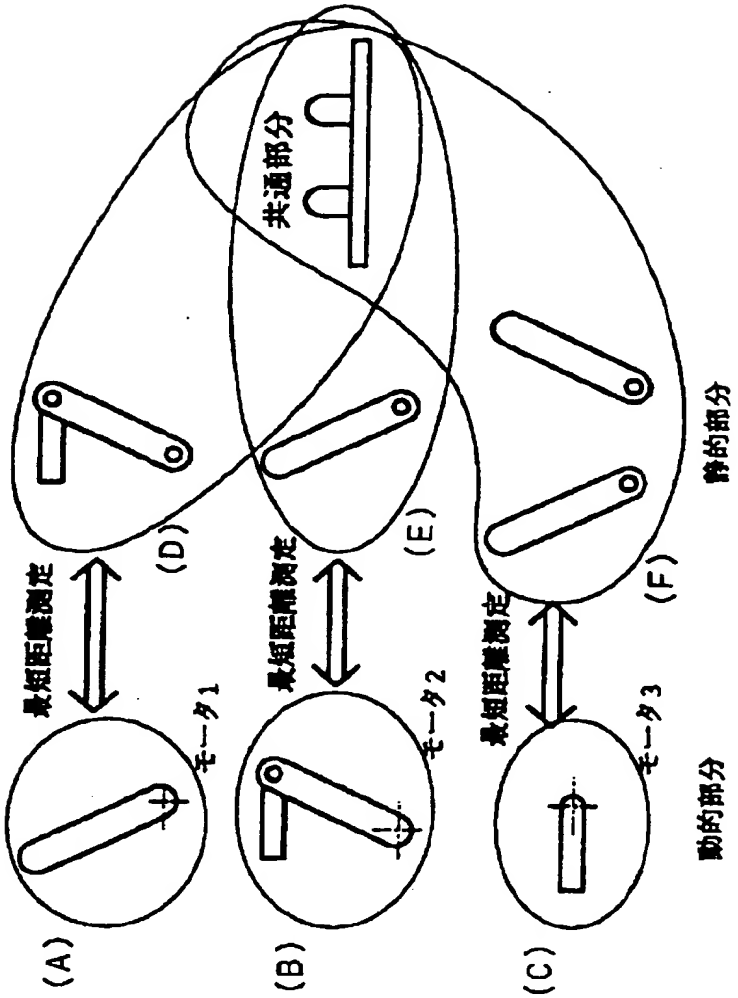
【図２６】



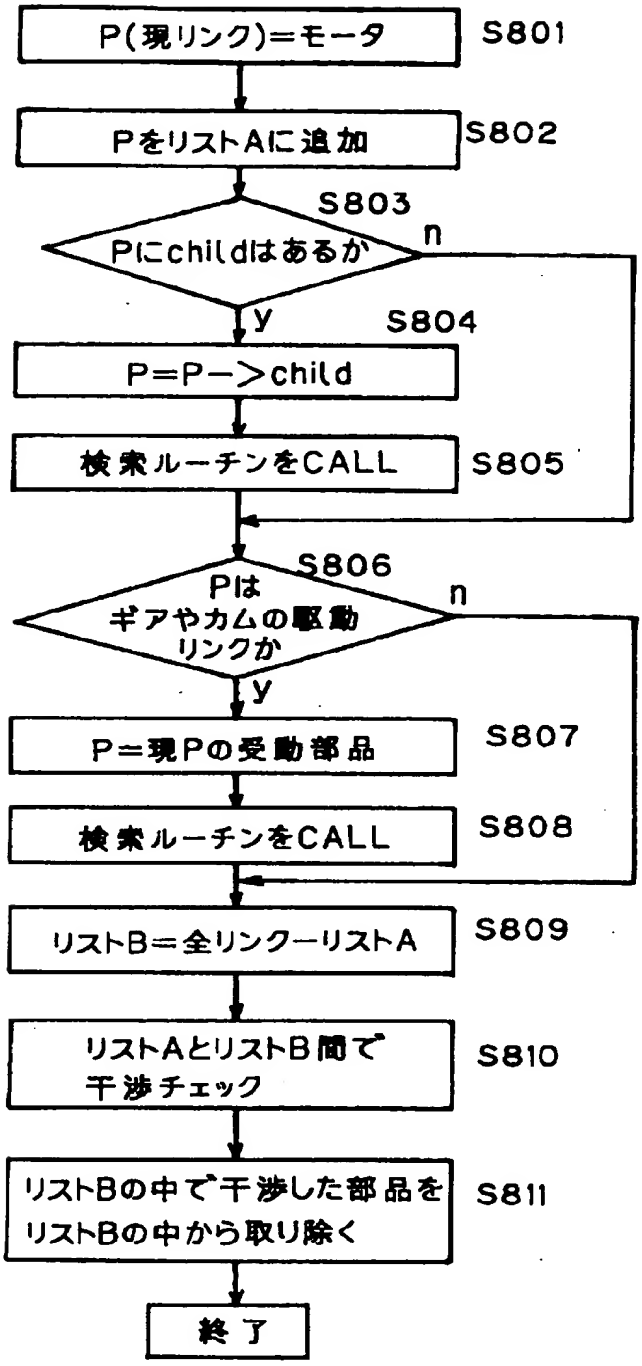
【図２７】



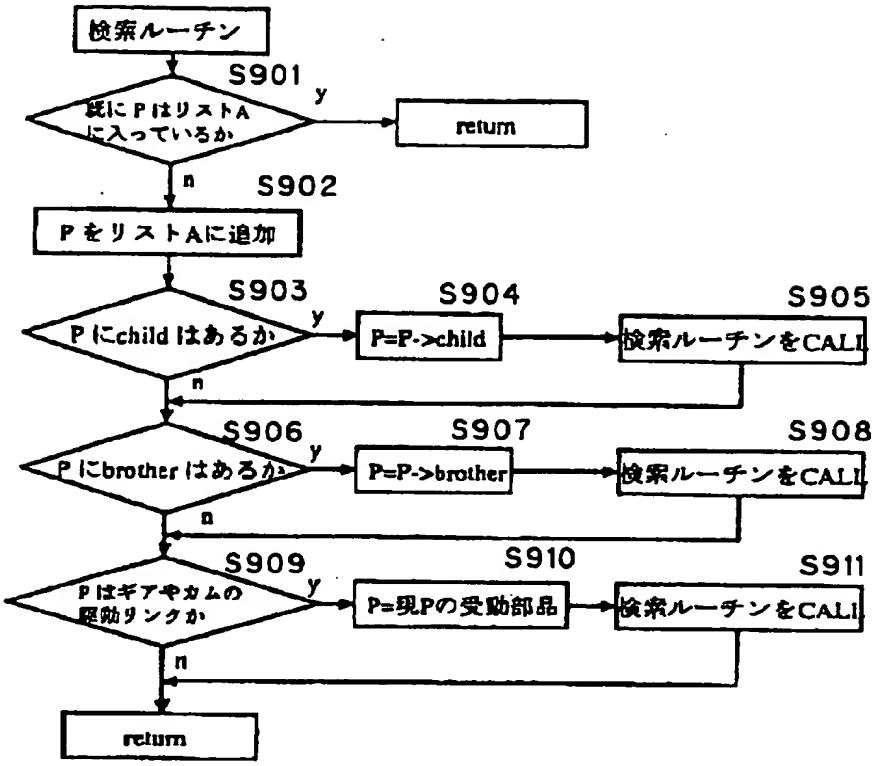
【図２８】



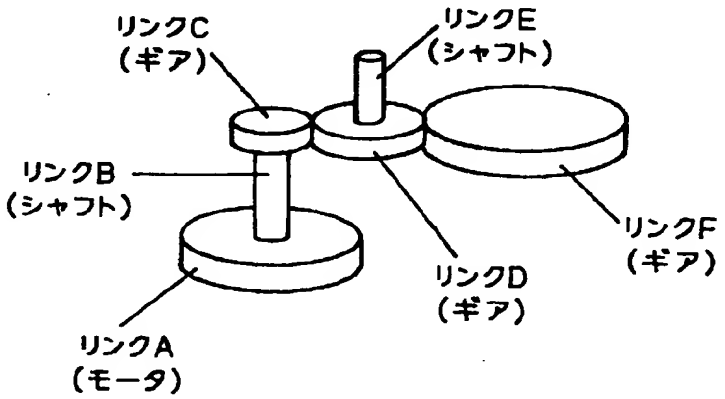
【図２９】


P(現リンク)=モータ
S801
PをリストAに追加
S802
S803
PにchildはあるかS
n
y
S804
P=P->child
検索ルーチンをCALL
S805
S806
Pは
ギアやカムの駆動
リンクか
n
y
P=現Pの受動部品
S807
検索ルーチンをCALL
S808
リストB=全リンクーリストA
S809
リストAとリストB間で
干渉チェック
S810
リストBの中で干渉した部品を
リストBの中から取り除く
S811
終了

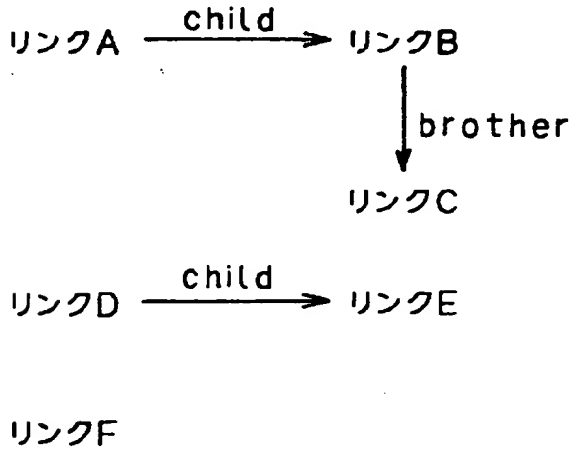
【図３０】


検索ルーチン
S901
既にPはリストAに入っているか
y
return
n
S902
Pをリストに追加
S903
Pにchildはあるか
y
S904
P=P->child
S905
検索ルーチンをCALL
n
S906
Pにbrotherはあるか
y
S907
P=P->brother
S908
検索ルーチンをCALL
n
S909
Pはギアやカムの駆動リンクか
y
S910
P=現Pの受動部品
S911
検索ルーチンをCALL
n
return

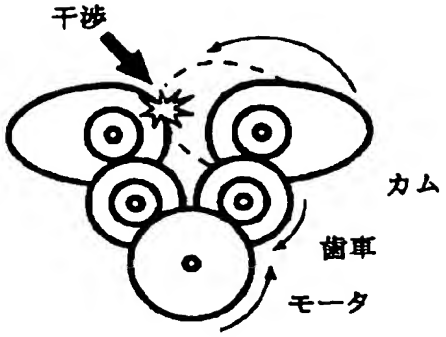
【図３１】


リンクC
(ギア)
リンクE
(シャフト)
リンクB
(シャフト)
リンクF
(ギア)
リンクD
(ギア)
リンクA
(モータ)

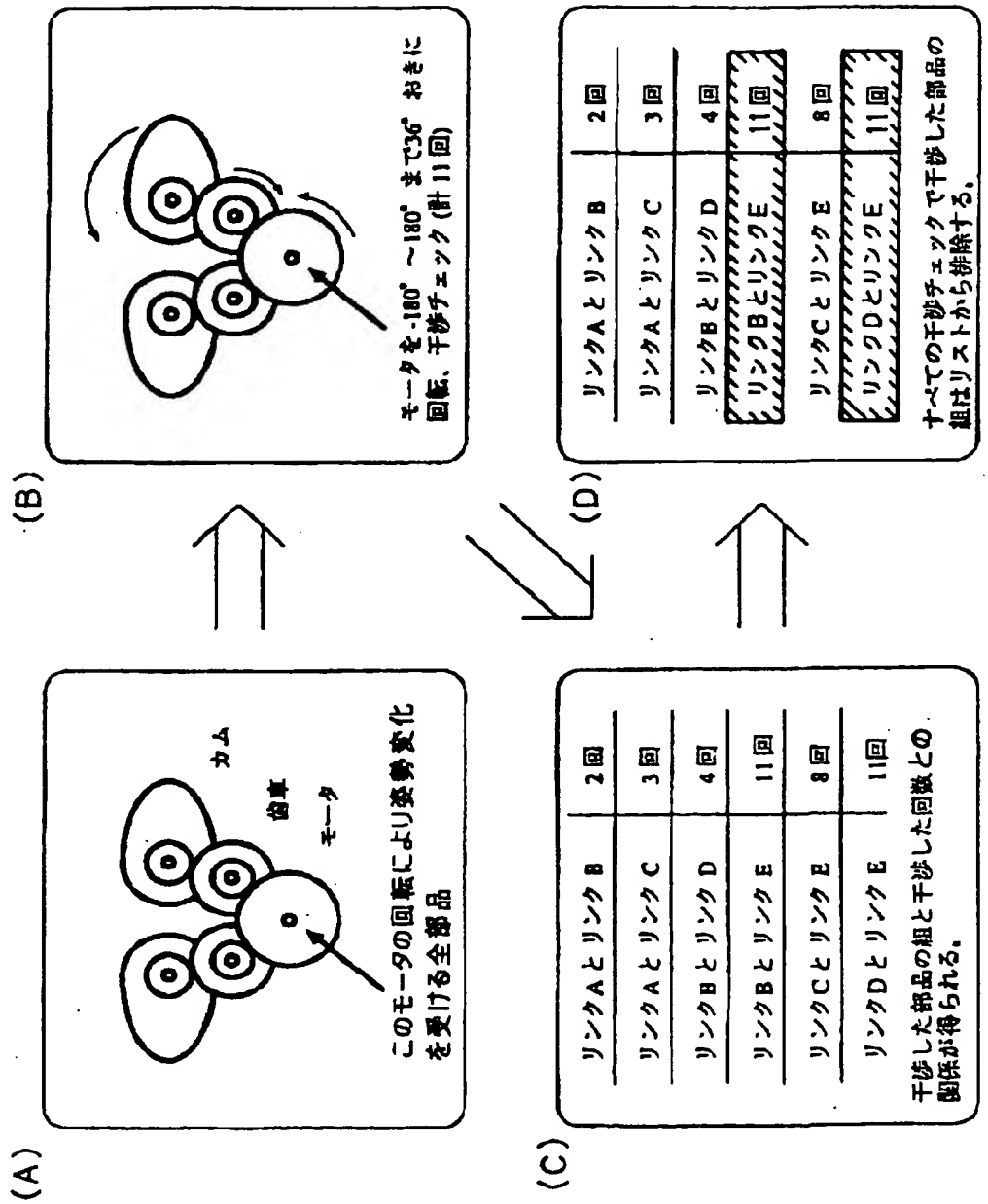
【図３２】


リンクA
child
リンクB
brother
リンクC
リンクD
child
リンクE
リンクF

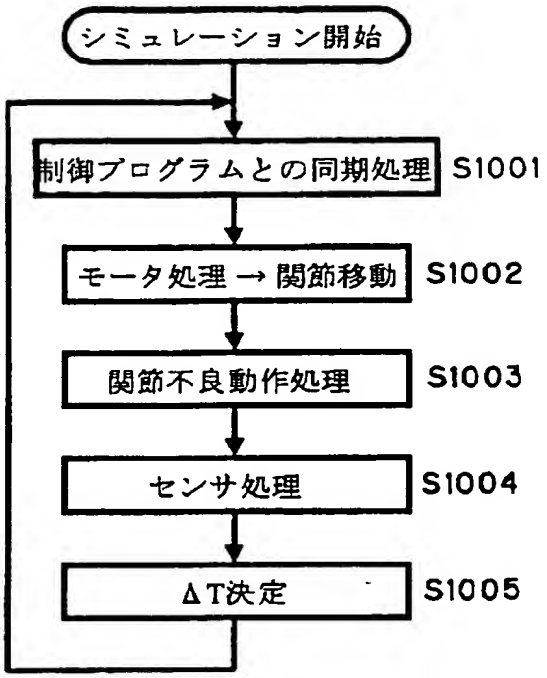
【図３３】


干渉
カム
歯車
モータ

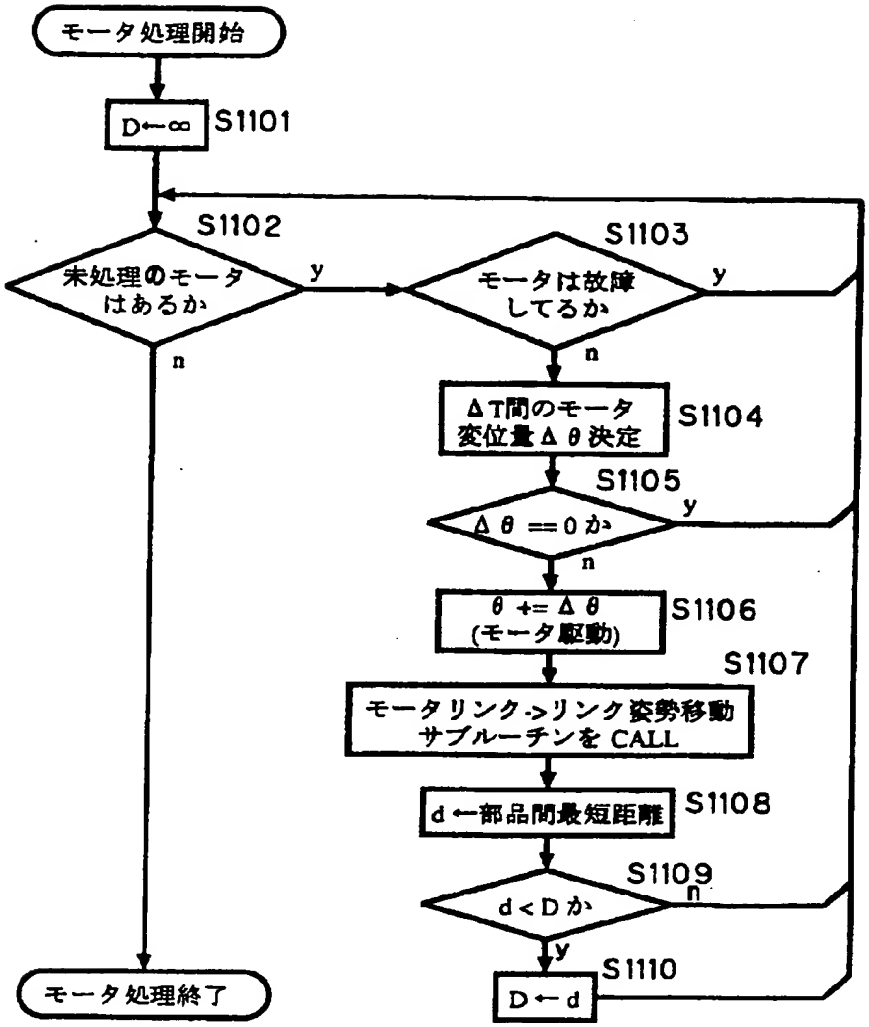
【図３４】


(A)
カム
歯車
モータ
このモータの回転により姿勢変化を受ける全部品
(B)
モータを-180°～180°まで36°おきに回転、干渉チェック(計11回)
(C)
リンクAとリンクB 2回
リンクAとリンクC 3回
リンクBとリンクD 4回
リンクBとリンクE 11回
リンクCとリンクE 8回
リンクDとリンクE 11回
干渉した部品の組と干渉した回数との関係が得られる。
(D)
リンクAとリンクB 2回
リンクAとリンクC 3回
リンクBとリンクD 4回
リンクBとリンクE 11回
リンクCとリンクE 8回
リンクDとリンクE 11回
すべての干渉チェックで干渉した部品の組はリストから排除する。

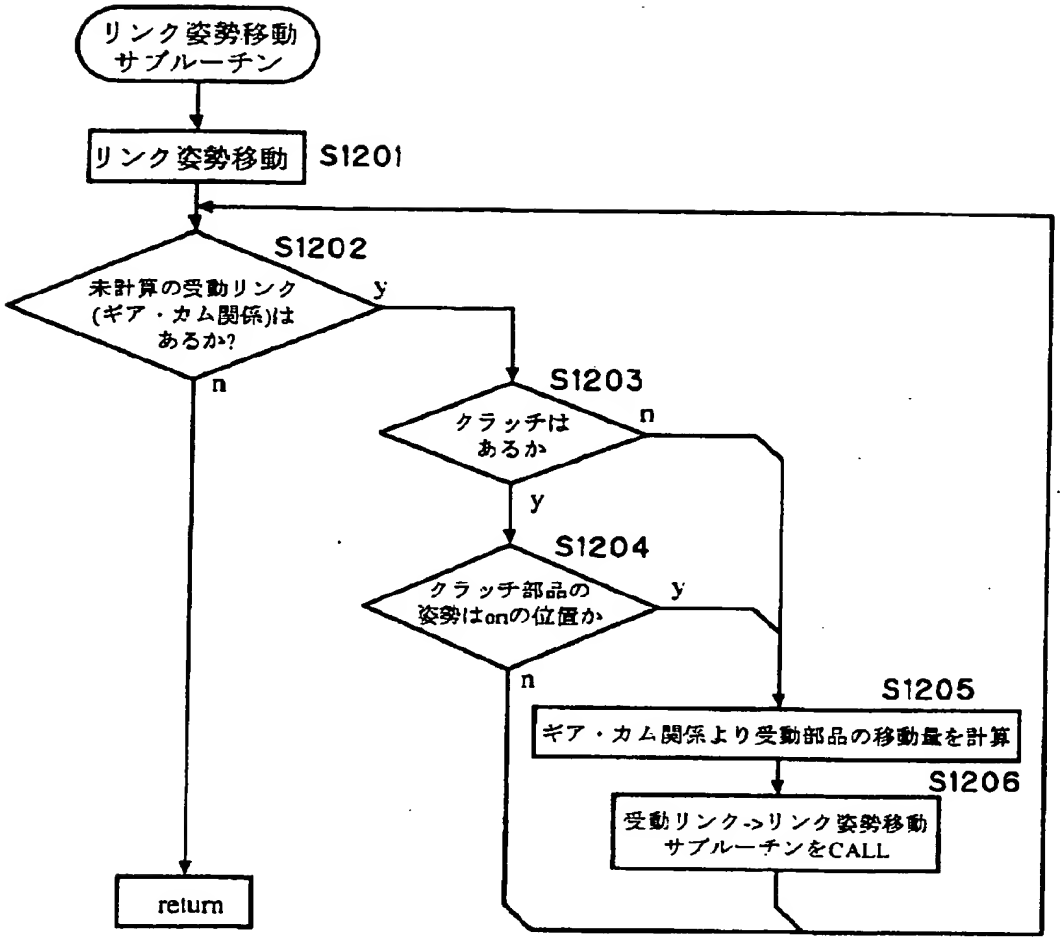
【図３５】


シミュレーション開始
制御プログラムとの同期処理
S1001
モータ処理 → 関節移動
S1002
関節不良動作処理
S1003
センサ処理
S1004
ΔT決定
S1005

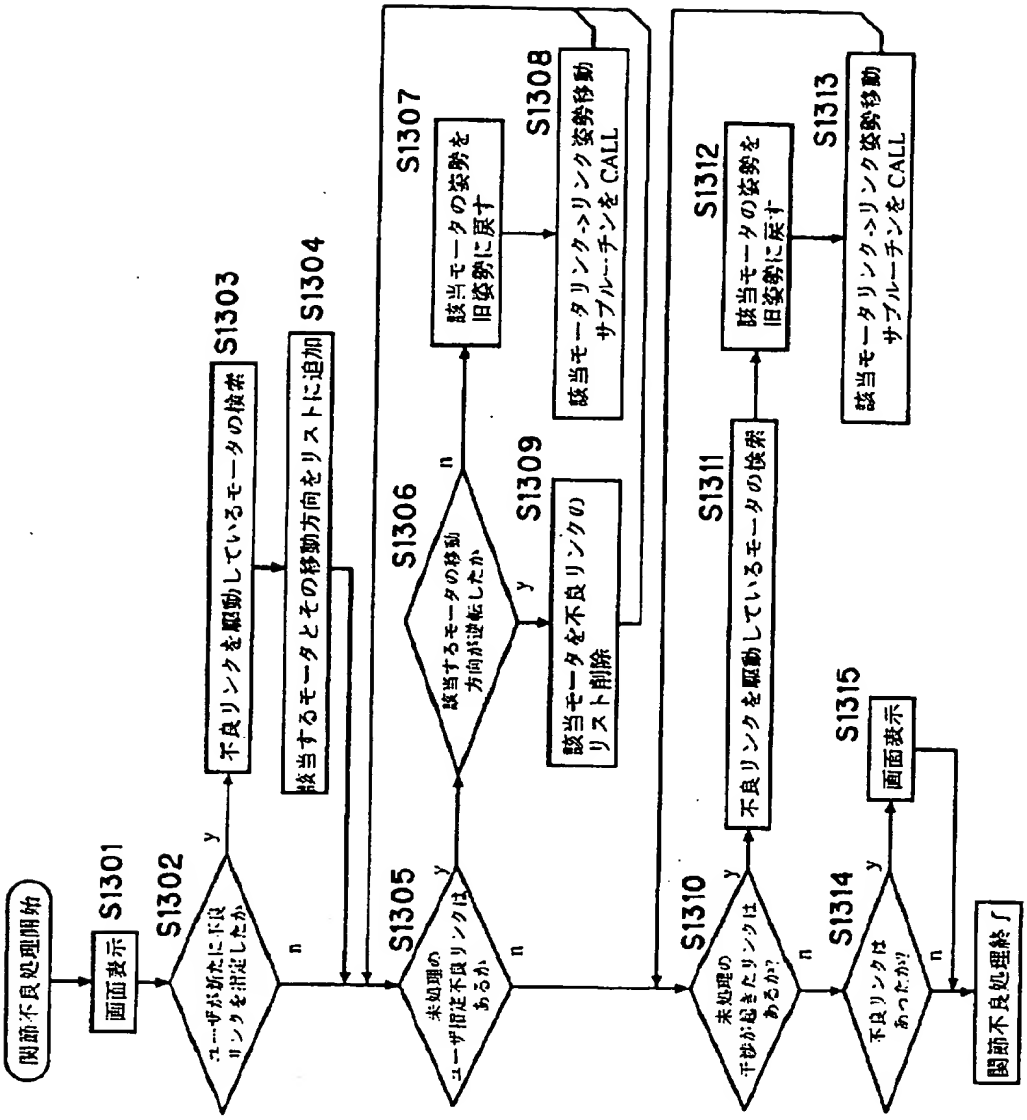
【図３６】


モータ処理開始
D←∞
S1101
S1102
未処理のモータはあるか
y
n
S1103
モータは故障してるか
y
n
ΔT間のモータ変位量Δθ決定
S1104
S1105
Δθ ==0か
y
n
θ +=Δθ (モータ駆動)
S1106
S1107
モータリンク->リンク姿勢移動サブルーチンを CALL
d←部品間最短距離
S1108
S1109
d<Dか
n
y
D←d
S1110
モータ処理終了

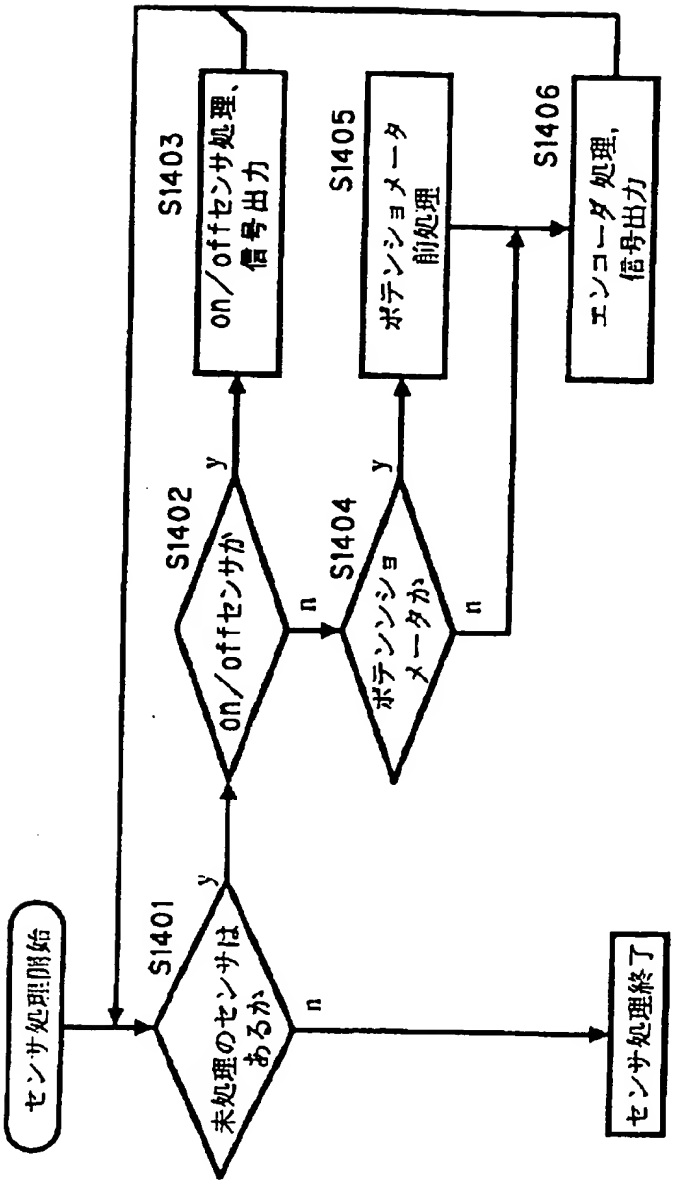
【図３７】


リンク姿勢移動
サブルーチン
リンク姿勢移動
S1201
S1202
未計算の受動リンク
(ギア・カム関係)は
あるか?
y
n
S1203
クラッチは
あるか
n
y
S1204
クラッチ部品の
姿勢はonの位置か
y
n
S1205
ギア・カム関係より受動部品の移動量を計算
S1206
受動リンク->リンク姿勢移動
サブルーチンをCALL
return

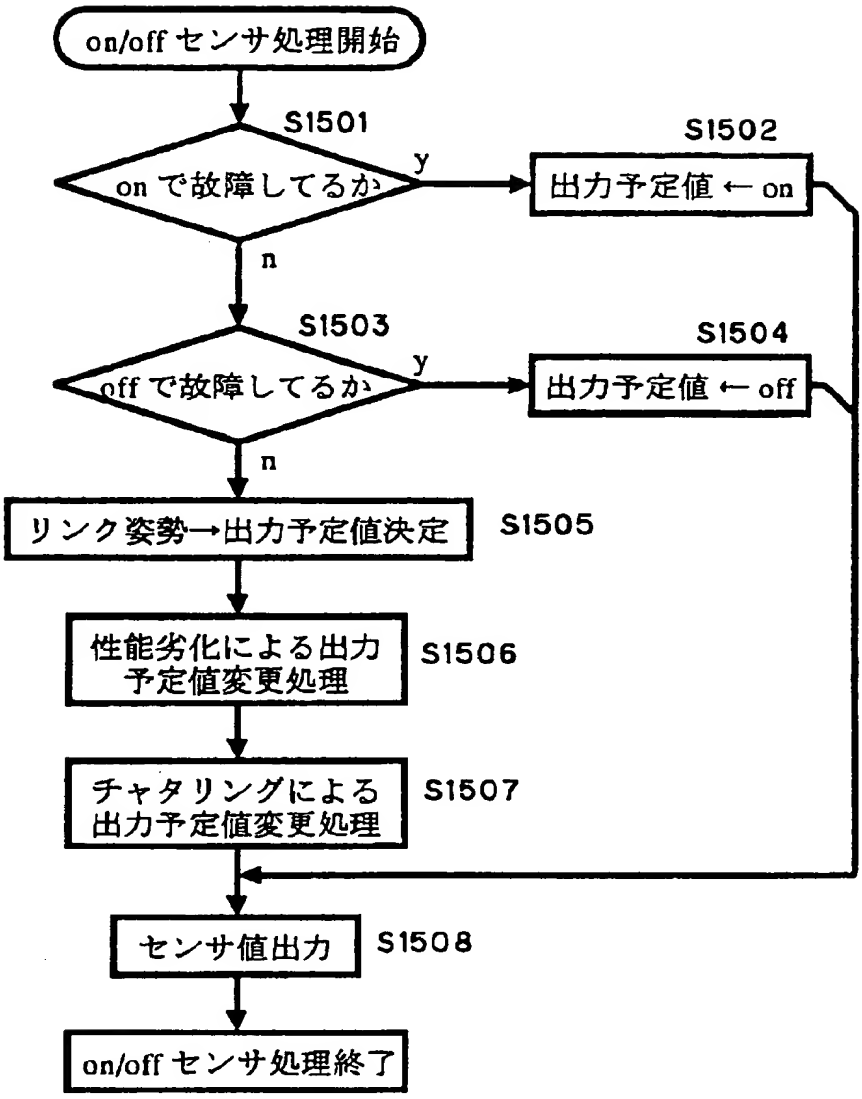
【図３８】


関節不良処理開始
S1301
画面表示
S1302
ユーザが新たに不良リンクを指定したか
y
n
S1303
不良リンクを駆動しているモータの検索
S1304
該当するモータとその移動方向をリストに追加
S1305
未処理のユーザ指定不良リンクはあるか
y
n
S1306
該当するモータの移動方向が逆転したか
y
n
S1307
該当モータの姿勢を旧姿勢に戻す
S1308
該当モータリンク->リンク姿勢移動サブルーチンをCALL
S1309
該当モータを不良リンクのリスト削除
S1310
未処理の干渉が起きたリンクはあるか?
y
n
S1311
不良リンクを駆動しているモータの検索
S1312
該当モータの姿勢を旧姿勢に戻す
S1313
該当モータリンク->リンク姿勢移動サブルーチンをCALL
S1314
不良リンクはあったか?
y
n
S1315
画面表示
関節不良処理終了

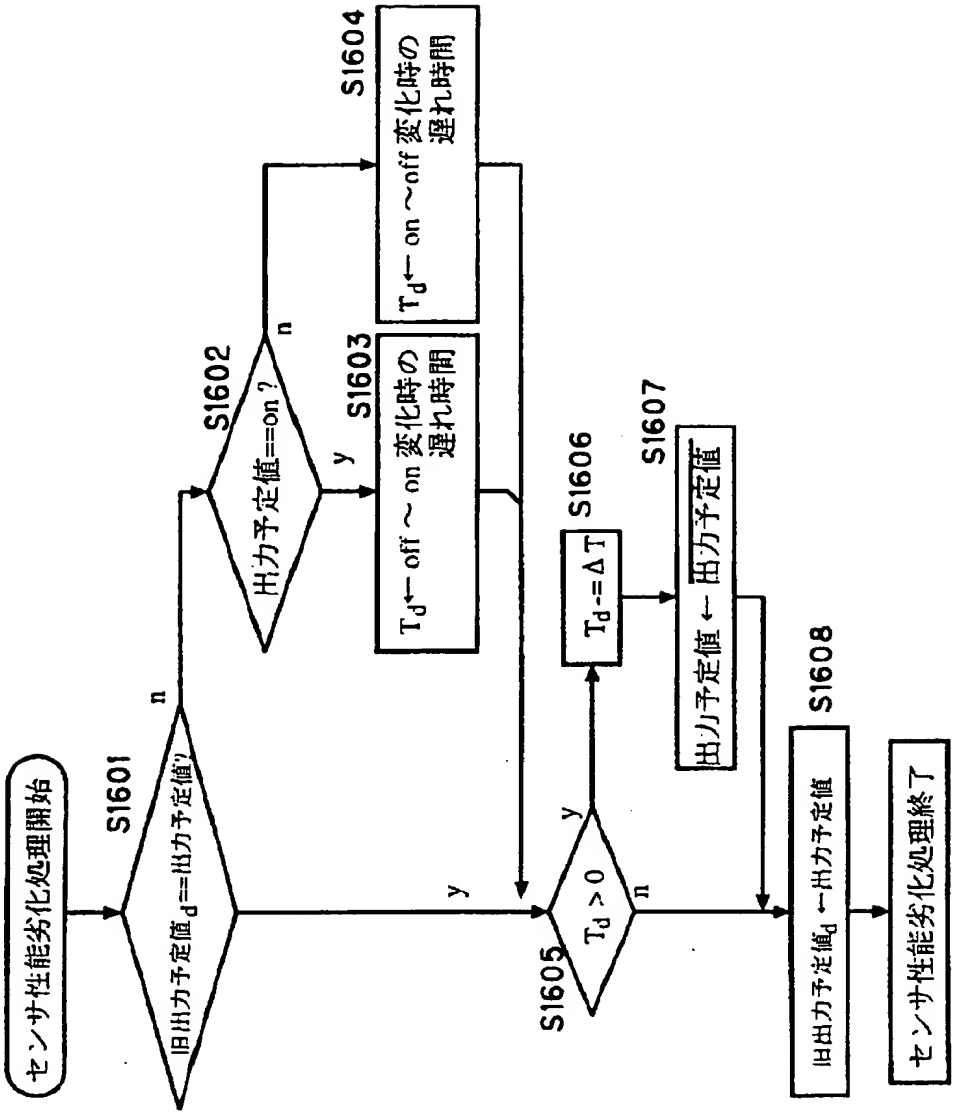
【図３９】


センサ処理開始
S1401
未処理のセンサは
あるか
y
n
S1402
on／offセンサか
y
n
S1403
on／offセンサ処理,
信号出力
S1404
ポテンショ
メータか
y
n
S1405
ポテンショメータ
前処理
S1406
エンコーダ処理,
信号出力
センサ処理終了

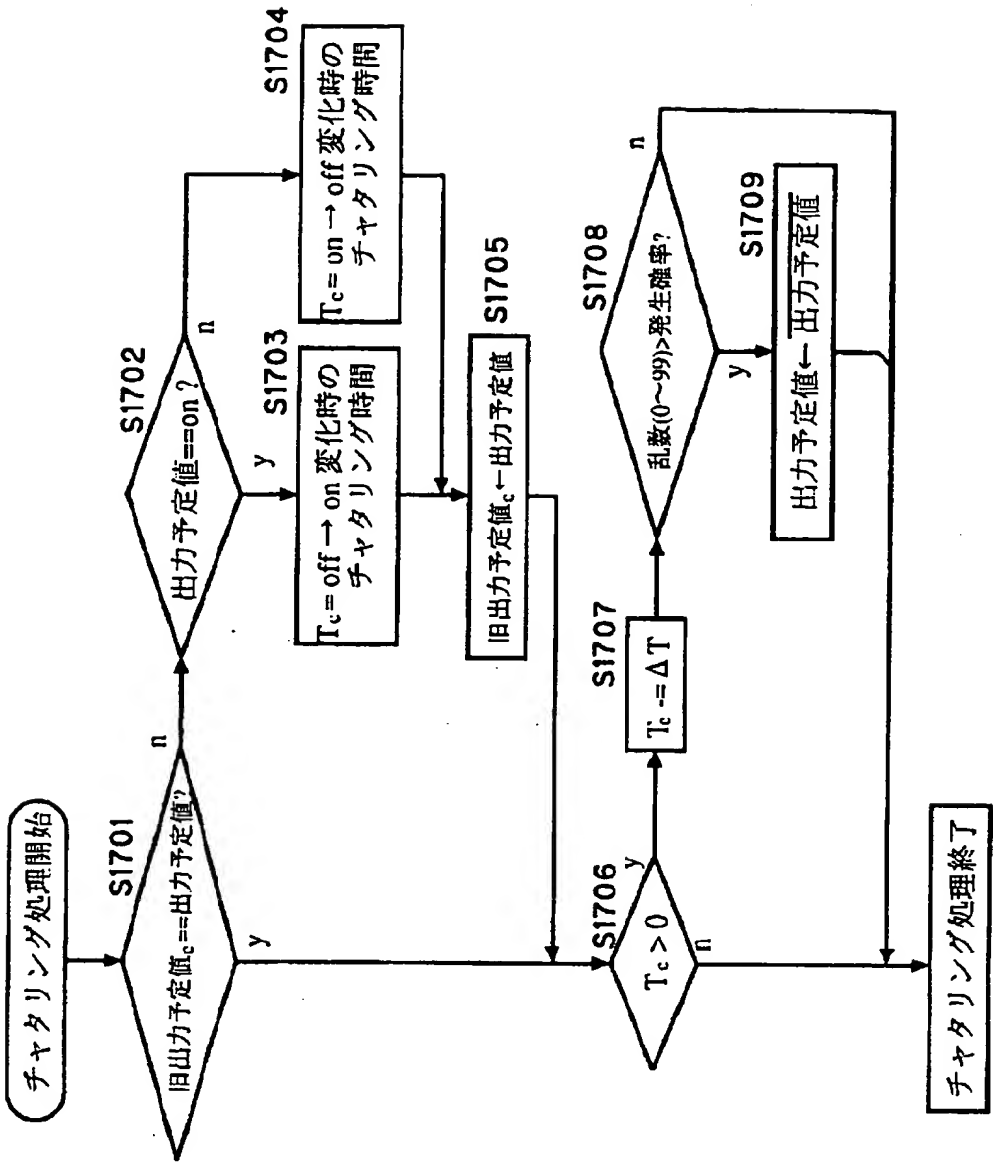
【図４０】


on/off センサ処理開始
S1501
on で故障してるか
y
S1502
出力予定値 ← on
n
S1503
off で故障してるか
y
S1504
出力予定値 ← off
n
リンク姿勢→出力予定値決定
S1505
性能劣化による出力予定値変更処理
S1506
チャタリングによる出力予定値変更処理
S1507
センサ値出力
S1508
on/off センサ処理終了

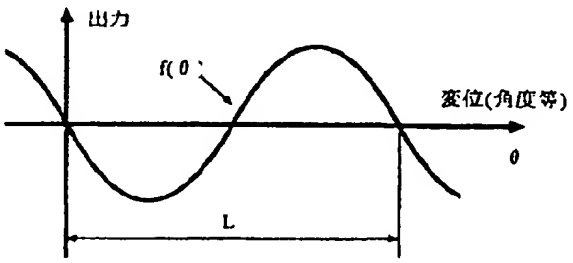
【図４１】


センサ性能劣化処理開始
S1601
旧出力予定値d==出力予定値'
n
y
S1602
出力予定値==on ?
n
y
S1603
Td← off ～ on 変化時の遅れ時間
S1604
Td← on ～off 変化時の遅れ時間
S1605
Td > 0
y
n
S1606
Td-=ΔT
S1607
出力予定値 ← 出力予定値
S1608
旧出力予定値d ←出力予定値
センサ性能劣化処理終了

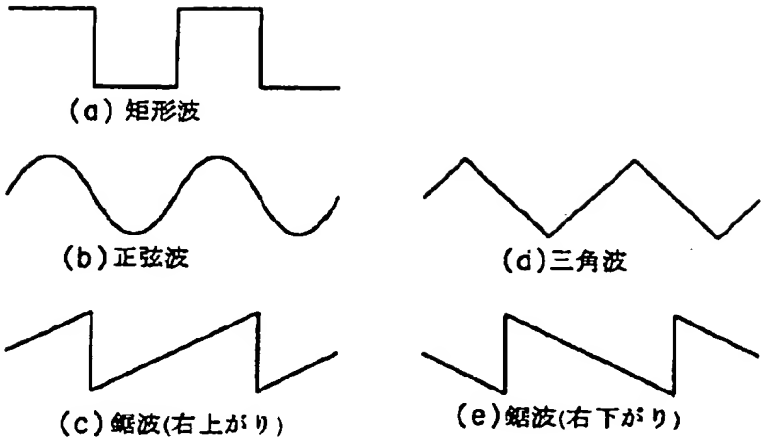
【図４２】


チャタリング処理開始
S1701
旧出力予定値c==出力予定値?
n
y
S1702
出力予定値==on ?
n
y
S1703
Tc= off → on 変化時のチャタリング時間
S1704
Tc= on → off 変化時のチャタリング時間
S1705
旧出力予定値c←出力予定値
S1706
Tc > 0
y
n
S1707
Tc -= ΔT
S1708
乱数(0~99)>発生確率?
n
y
S1709
出力予定値← 出力予定値
チャタリング処理終了

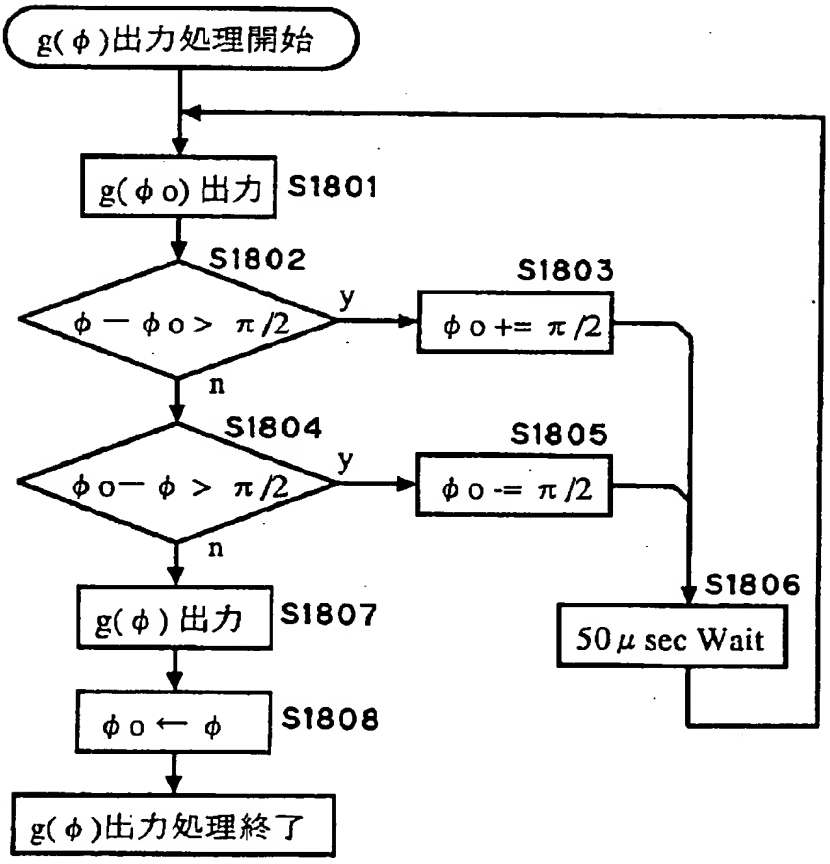
【図４３】


出力
f(θ)
変位(角度等)
θ
L

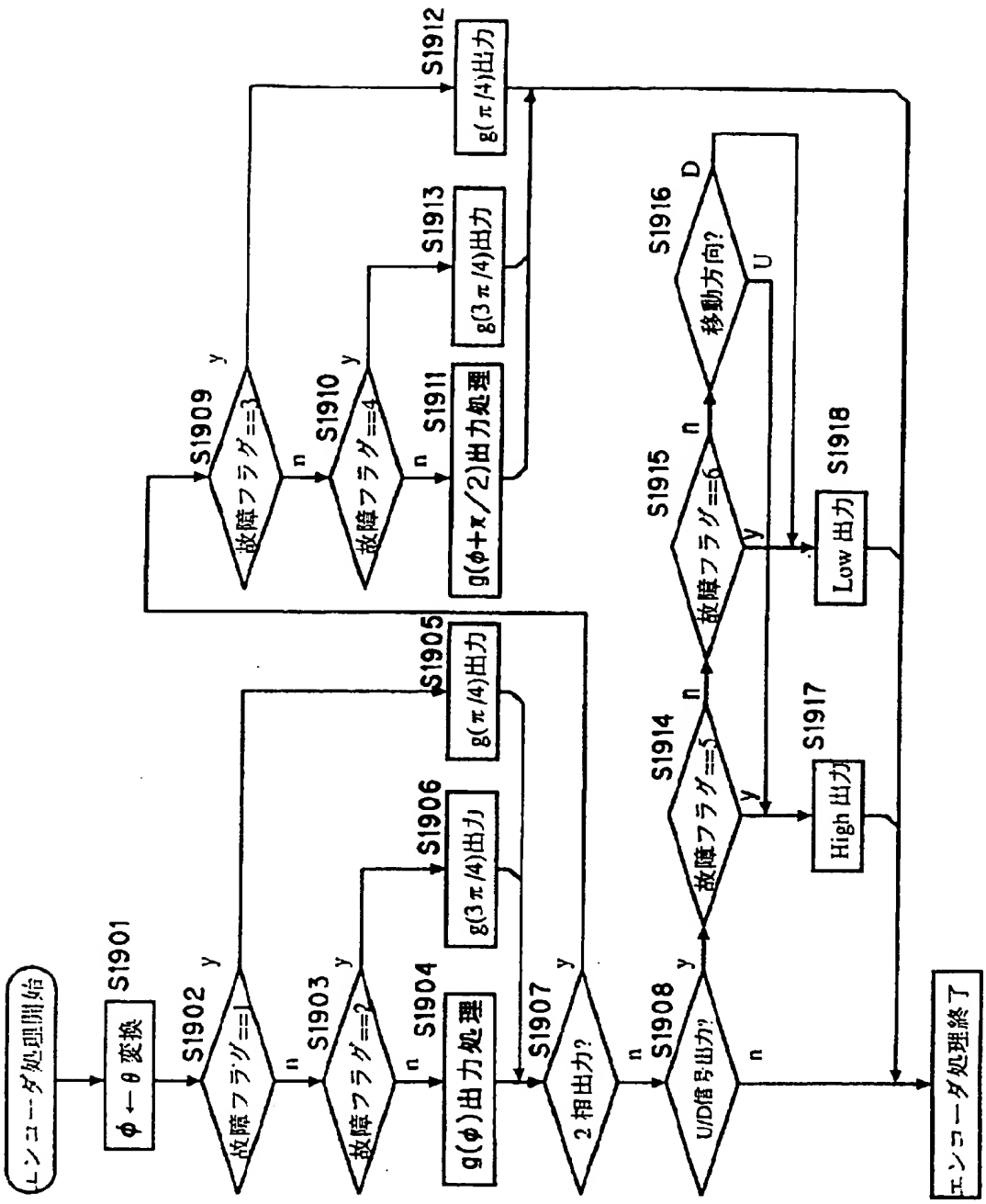
【図４４】


(a) 矩形波
(b)正弦波
(d)三角波
(c)鋸波(右上がり)
(e)鋸波(右下がり)

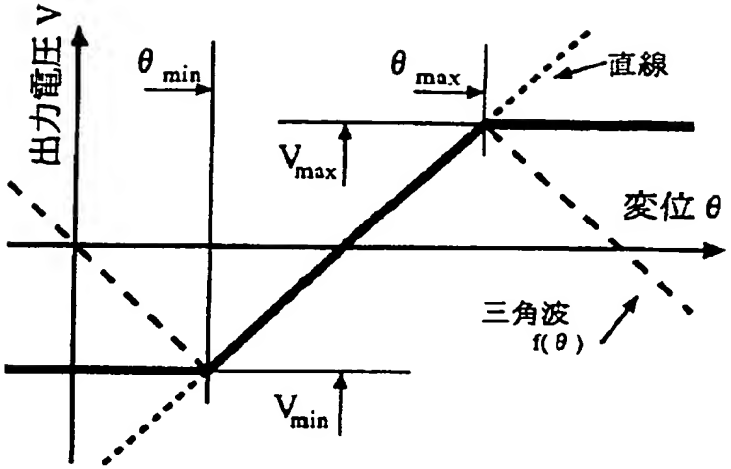
【図４５】


g(φ)出力処理開始
g(φo)出力
S1801
S1802
φ − φo > π/2
y
n
S1803
φo += π/2
S1804
φo − φ > π/2
y
n
S1805
φo -= π/2
S1806
50μsec Wait
g(φ)出力
S1807
φo ← φ
S1808
g(φ)出力処理終了

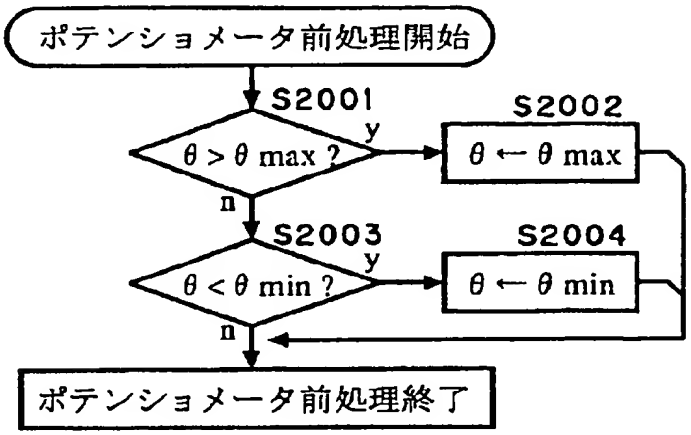
【図４６】


エンコーダ処理開始
S1901
φ←θ変換
S1902
故障フラグ==1
S1903
故障フラグ==2
S1904
g(φ)出力処理
S1905
g(π/4)出力
S1906
g(3π/4)出力
S1907
2相出力?
S1908
U/D信号出力?
S1909
故障フラグ==3
S1910
故障フラグ==4
S1911
g(φ+π/2)出力処理
S1912
g(π/4)出力
S1913
g(3π/4)出力
S1914
故障フラグ==5
S1915
故障フラグ==6
S1916
移動方向?
S1917
High出力
S1918
Low出力
エンコーダ処理終了
y
n
D
U

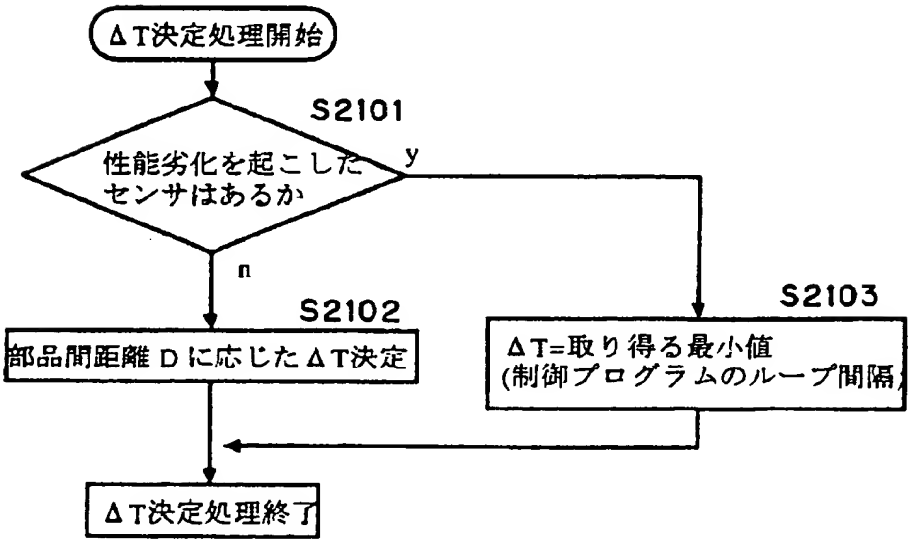
【図４７】


出力電圧 V
θ min
θ max
直線
Vmax
変位 θ
三角波
f(θ)
Vmin

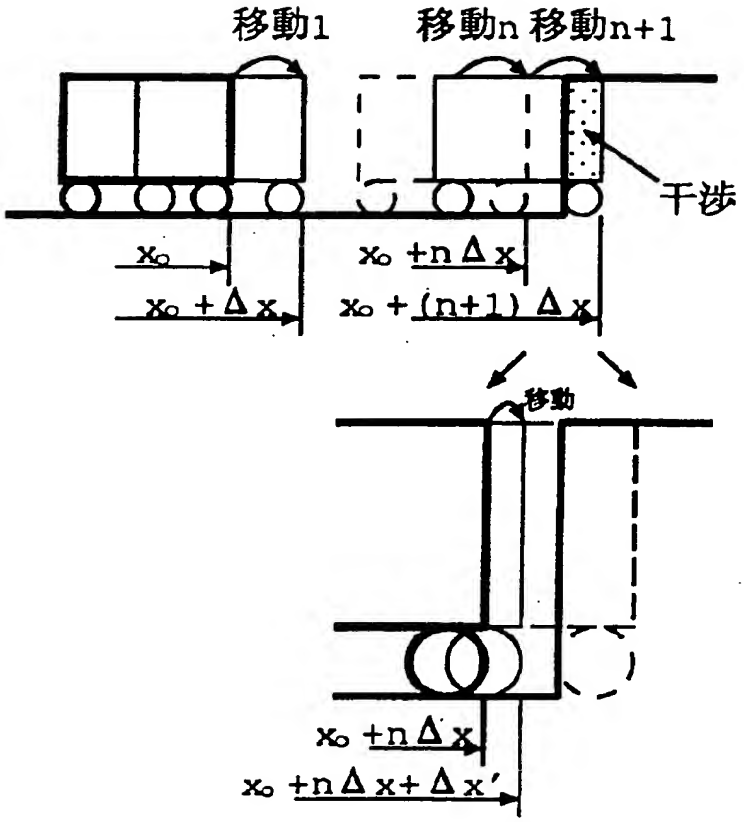
【図４８】


ポテンショメータ前処理開始
S2001
θ > θ max ?
y
n
S2002
θ ← θ max
S2003
θ < θ min ?
y
n
S2004
θ ← θ min
ポテンショメータ前処理終了

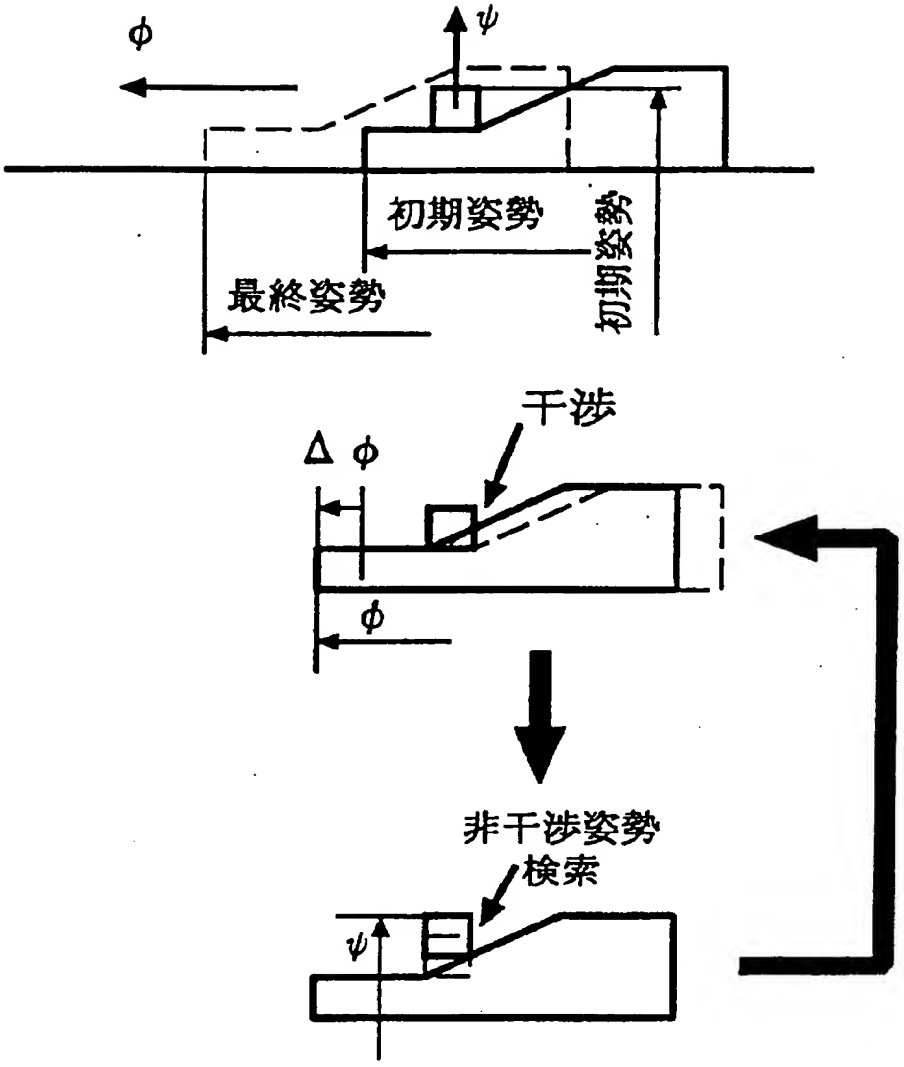
【図４９】


ΔT決定処理開始
S2101
性能劣化を起こした
センサはあるか
y
n
S2102
部品間距離 D に応じたΔT決定
S2103
ΔT=取り得る最小値
(制御プログラムのループ間隔)
ΔT決定処理終了

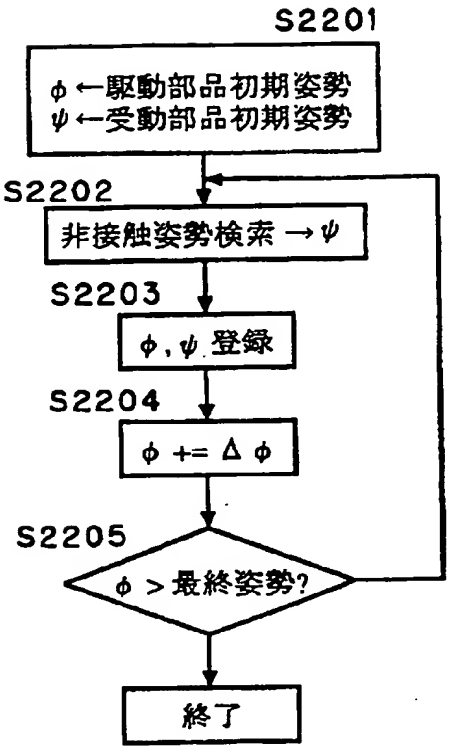
【図５０】



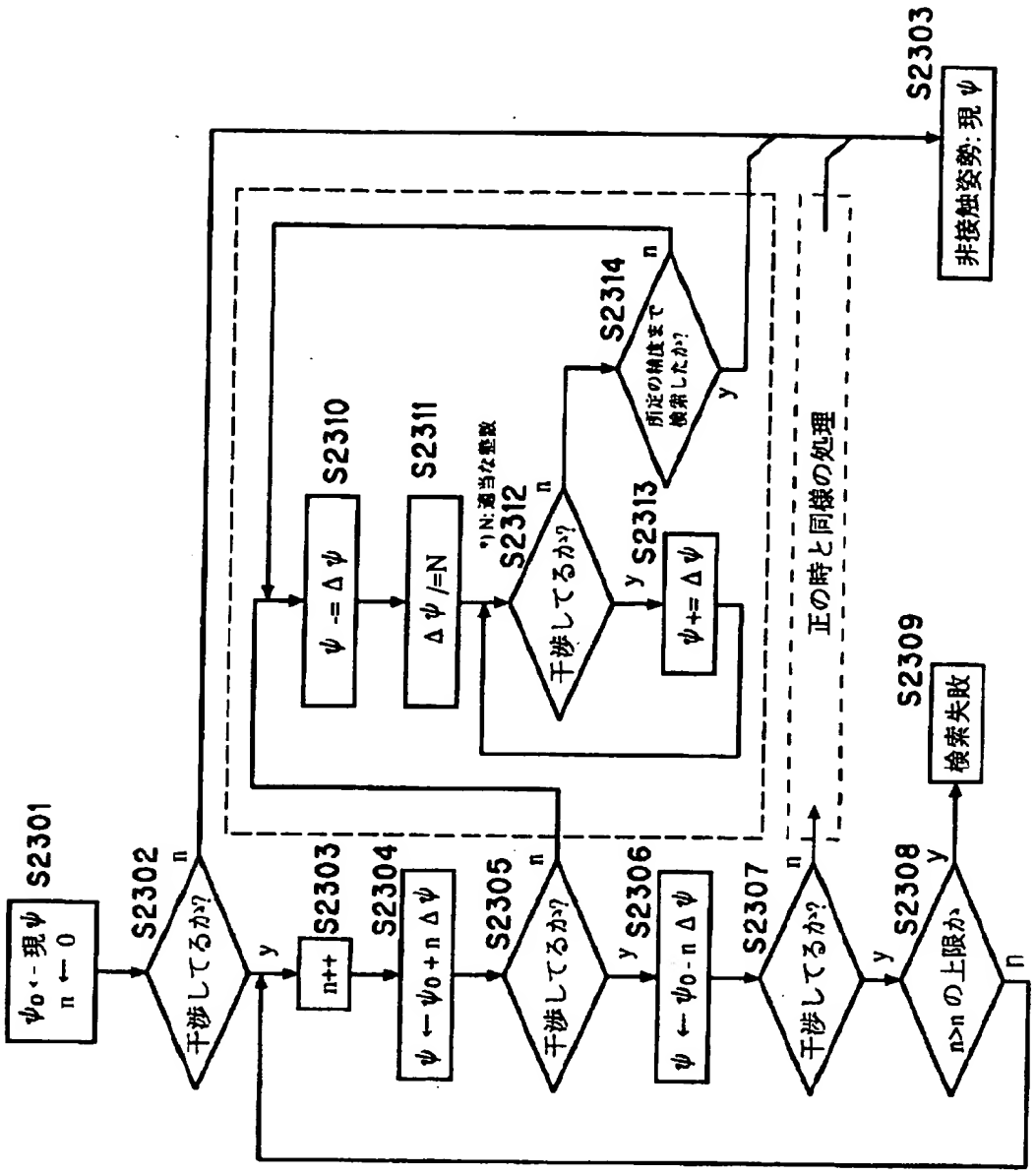
【図５１】



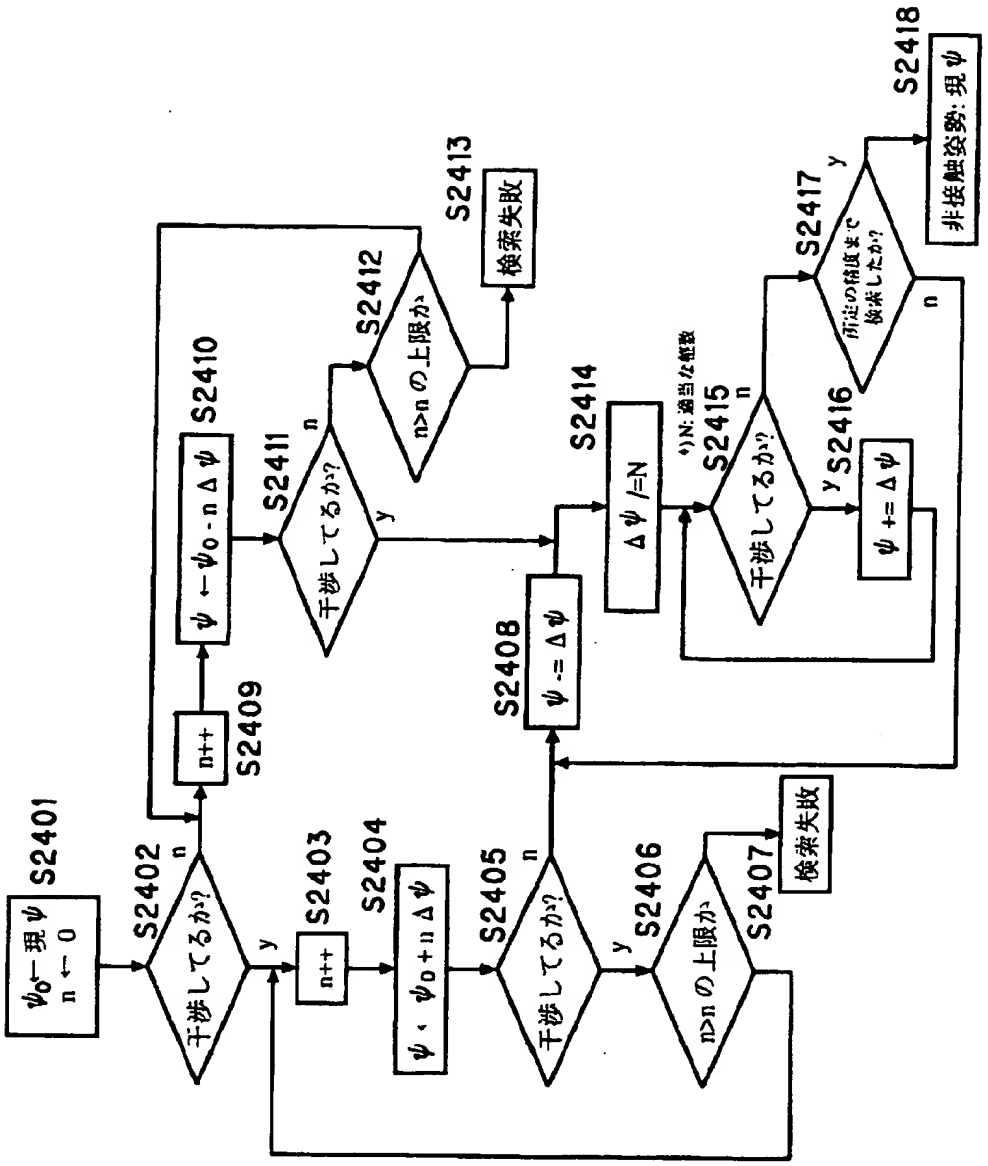
【図５２】


S2201
φ←駆動部品初期姿勢
ψ←受動部品初期姿勢
S2202
非接触姿勢検索→ψ
S2203
φ,ψ登録
S2204
φ += Δφ
S2205
φ > 最終姿勢?
終了

【図５３】


S2301
ψ0←現ψ
n←0
S2302
干渉してるか?
n
y
S2303
n++
S2304
ψ←ψ0+nΔψ
S2305
干渉してるか?
n
y
S2306
ψ←ψ0-nΔψ
S2307
干渉してるか?
n
y
S2308
n>nの上限か
y
n
S2309
検索失敗
S2303
非接触姿勢:現ψ
正の時と同様の処理
S2310
ψ-=Δψ
S2311
Δψ/=N
"1"N:適当な整数
S2312
干渉してるか?
n
y
S2313
ψ+=Δψ
S2314
所定の精度まで検索したか?
n
y

【図５４】


S2401
ψ0←現ψ
n←0
S2402
干渉してるか?
n
y
S2403
n++
S2404
ψ←ψ0+nΔψ
S2405
干渉してるか?
n
y
S2406
n>nの上限か
S2407
検索失敗
S2408
ψ-=Δψ
S2409
n++
S2410
ψ←ψ0-nΔψ
S2411
干渉してるか?
n
y
S2412
n>nの上限か
S2413
検索失敗
S2414
Δψ/=N
')N:適当な値数
S2415
干渉してるか?
n
y
S2416
ψ+=Δψ
S2417
所定の精度まで検索したか?
y
n
S2418
非接触姿勢:現ψ

【図５５】